



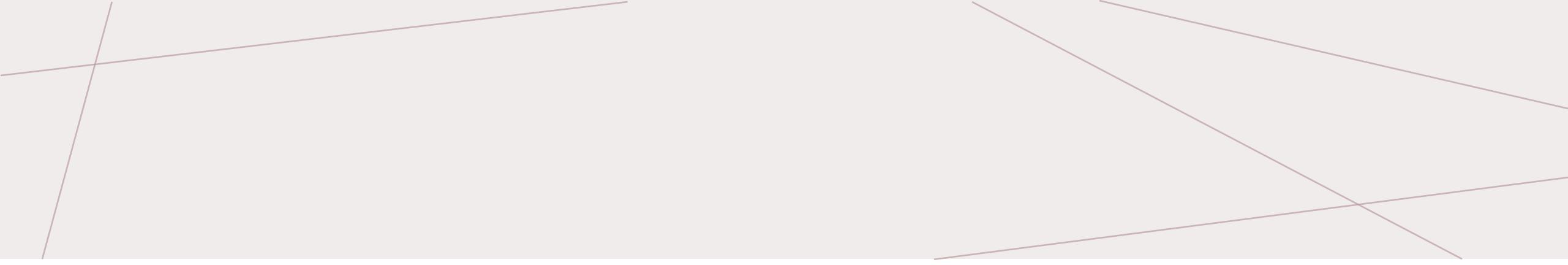
DJANGO

SZABLONY

PLAN

- **Szablony**
- **Języki szablonów**
 - Języki szablonów html dla Pythona
- **Język szablonów Django**
 - Elementy składniowe języka
 - Zmienne
 - Tagi
 - Filtry
 - Parametry filtrów
 - Komentarze
 - Przykłady zastosowań
 - Treści statyczne
 - Warunki
 - Pętle
 - Dynamiczne linkowanie
 - Dziedziczenie
- **Dodatki**
 - Własne tagi i filtry

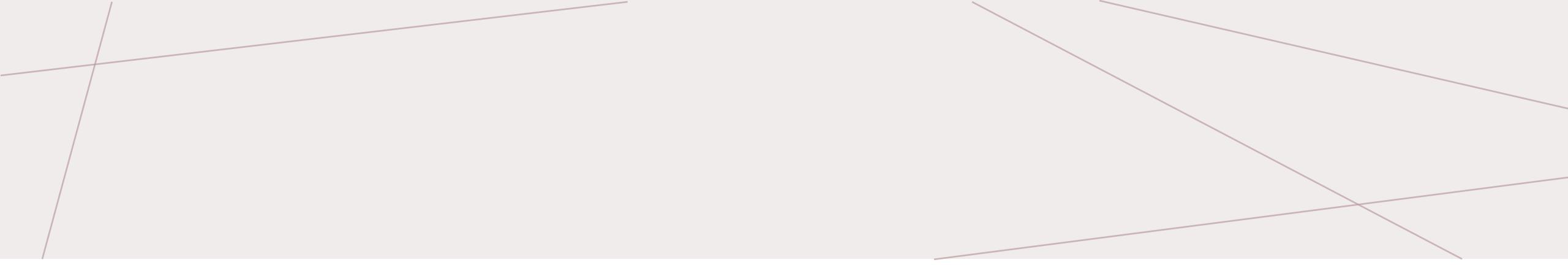
SZABLONY



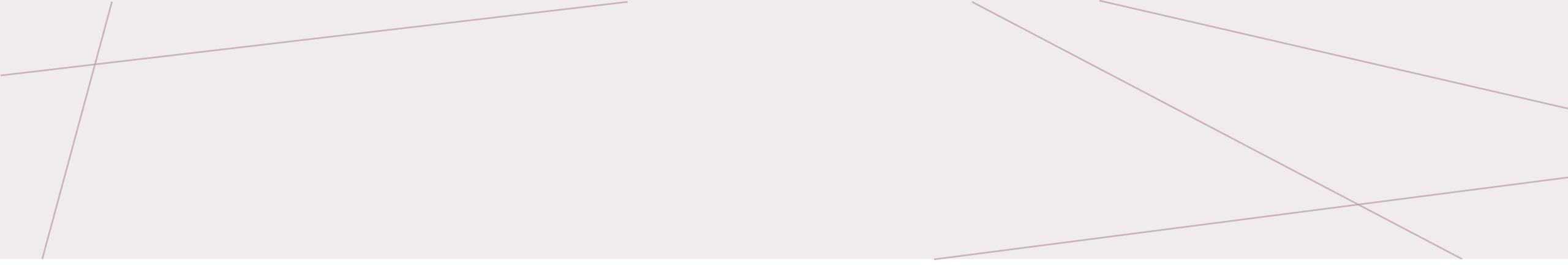
Szablon - plik tekstowy definiujący sposób wyświetlania danych, buduje tzw. **warstwę prezentacji aplikacji**.

*Szablon może służyć do wygenerowania dowolnego formatu tekstu. Treść szablonu generowana jest automatycznie za pomocą instrukcji tzw. **języka szablonów**.*

JEZYKI SZABLONOW



Język szablonów służy do **automatycznego generowania treści**. Format automatycznie wygenerowanej treści zależy od użytego języka szablonów. Do najpopularniejszych języków szablonów należą języki generujące treść w formacie *xml (XML-based template language)* oraz języki generujące treści w formacie *html (HTML-based template language)*.



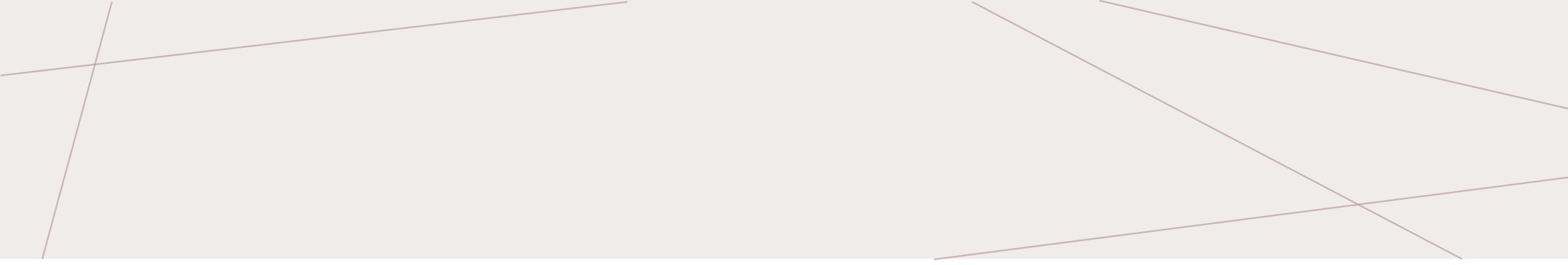
Każdy język programowania posiada swoje, dedykowane języki szablonów.

Dla Pythona:

- najpopularniejsze języki szablonów przeznaczone do generowania plików XML to m.in. *Genshi XML* i *Zope's TAL*
- najpopularniejsze języki szablonów generujące pliki html dla Pythona to: *Jinja*, *Mako*, *Genshi*, *Mustache*. *Django* posiada swój własny język szablonów.

Język szablonów Django używamy głównie do generowania plików html, ale (jak większość języków szablonów) może być z powodzeniem użyty do generowania dowolnego formatu tekstowego (email, csv, json, ...).

Szersze omówienie języków szablonów dla Pythona można znaleźć [tutaj](#)



Logika wyrażona za pomocą języka szablonów powinna dotyczyć wyłącznie zagadnień związanych z prezentacją danych. Za pomocą języka szablonów nie należy implementować logiki biznesowej aplikacji.

*JEZYKI
SZABLONOW HTML
DLA PYTHONA*

Project Links

[Donate](#)
[PyPI Releases](#)
[Source Code](#)
[Issue Tracker](#)
[Website](#)
[Twitter](#)
[Chat](#)

Quick search


Stop searching. Start doing.
We simplify search for complex enterprises. [Ask for a 30 Day Free Trial](#)
Ad by EthicalAds · Monetize your site



Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.

Contents:

- [Introduction](#)
 - [Installation](#)
- [API](#)
 - [Basics](#)
 - [High Level API](#)
 - [Autoescaping](#)
 - [Notes on Identifiers](#)
 - [Undefined Types](#)
 - [The Context](#)
 - [Loaders](#)
 - [Bytecode Cache](#)
 - [Async Support](#)
 - [Policies](#)
 - [Utilities](#)
 - [Exceptions](#)
 - [Custom Filters](#)
 - [Custom Tags](#)

JINJA



Hyperfast and lightweight templating for the Python platform.

[Home](#) | [Community](#) | [Documentation](#) | [Download](#)

Mako Templates for Python

Mako is a template library written in Python. It provides a familiar, non-XML syntax which compiles into Python modules for maximum performance. Mako's syntax and API borrows from the best ideas of many others, including Django and Jinja2 templates, Cheetah, Myghty, and Genshi. Conceptually, Mako is an embedded Python (i.e. Python Server Page) language, which refines the familiar ideas of componentized layout and inheritance to produce one of the most straightforward and flexible models available, while also maintaining close ties to Python calling and scoping semantics.

Mako is used by [reddit.com](#) where it delivers over [one billion page views per month](#). It is the default template language included with the [Pylons](#) and [Pyramid](#) web frameworks.

Nutshell:

```
<%inherit file="base.html"/>
<%
    rows = [[v for v in range(0,10)] for row in range(0,10)]
%>





<%def name="makerow(row)">
    <tr>
        % for name in row:
            <td>${name}</td> \
        % endfor
    </tr>
<%def>
```

MAKO

Philosophy:



Szukaj

Zaloguj się | Ustawienia | Pomoc/Przewodnik | O systemie Trac

Home Trac Trac Hacks Genshi Babel Bitten

Wiki

Historia

Plan prac

Przeglądaj repozytorium

Zobacz zgłoszenia

Szukaj

wiki: WikiStart

Strona poczatkowa | Indeks | Historia

Genshi

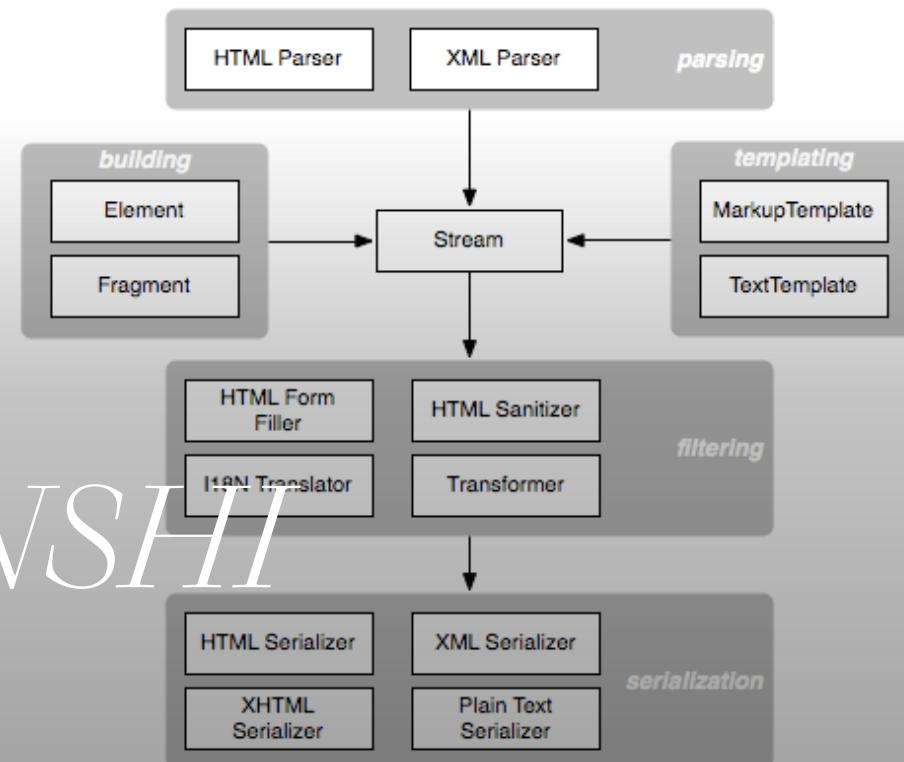
Python toolkit for generation of output for the web

Genshi is a Python library that provides an integrated set of components for parsing, generating, and processing HTML, XML or other textual content for output generation on the web.

The main feature is a [template language](#) that is smart about markup: unlike conventional template languages that only deal with bytes and (if you're lucky) characters, Genshi knows the difference between tags, attributes, and actual text nodes, and uses that knowledge to your advantage. For example:

- Intelligent automatic escaping greatly reduces the risk of opening up your site to [cross-site scripting](#) attacks (XSS).
- [Template directives](#) are often less verbose than those in most other template languages, as they can be attached directly to the elements they act upon.
- [Independence from a specific serialization format](#) lets you instantly switch between generating well-formed HTML 4.01 and XHTML 1.0 (or other formats).
- [Stream-based filtering](#) allows you to apply various transformations to a template as it is being processed, without having to parse and serialize the output again.
- [Match templates](#) let you enforce a common structure on template output, and more. This, in combination with [XInclude](#) support, is used instead of the more rigid inheritance feature commonly found in other template languages.

For those cases where you don't want to generate markup, Genshi also provides a simple [text-based template language](#).





Logic-less templates.

Available in [Ruby](#), [JavaScript](#), [Python](#),
[Erlang](#), [Elixir](#), [PHP](#), [Perl](#), [Raku](#), [Objective-C](#),
[Java](#), [C#/.NET](#), [Android](#), [C++](#), [CFEngine](#),
[Go](#), [Lua](#), [ooc](#), [ActionScript](#), [ColdFusion](#),
[Scala](#), [Clojure\[Script\]](#), [Clojure](#), [Fantom](#),
[CoffeeScript](#), [D](#), [Haskell](#), [XQuery](#), [ASP](#), [Io](#),
[Dart](#), [Haxe](#), [Delphi](#), [Racket](#), [Rust](#), [OCaml](#),
[Swift](#), [Bash](#), [Julia](#), [R](#), [Crystal](#), [Common Lisp](#),
[Nim](#), [Pharo](#), [Tcl](#), [C](#), [ABAP](#), [Elm](#), [Kotlin](#), and
for [SQL](#) **MUSTACHE**

Works great with TextMate Vim Emacs

Documentation

Search 3.2 documentation



The Django template language

This document explains the language syntax of the Django template system. If you're looking for a more technical perspective on how it works and how to extend it, see [The Django template language: for Python programmers](#).

Django's template language is designed to strike a balance between power and ease. It's designed to feel comfortable to those used to working with HTML. If you have any exposure to other text-based template languages, such as [Smarty](#) or [Jinja2](#), you should feel right at home with Django's templates.



Philosophy

If you have a background in programming, or if you're used to languages which mix programming code directly into HTML, you'll want to bear in mind that the Django template system is not simply Python embedded into HTML. This is by design: the template system is meant to express presentation, not program logic.

The Django template system provides tags which function similarly to some programming constructs – an `if` tag for boolean tests, a `for` tag for looping, etc. – but these are not simply executed as the corresponding Python code, and the template system will not execute arbitrary Python expressions. Only the tags, filters and syntax listed below are supported by default (although you can add [your own extensions](#) to the template language as needed).

JĘZYK SZABLONOW D J A N G O

Support Django!



WashDrop donated to the Django Software Foundation to support Django development. [Donate today!](#)

Contents

- [The Django template language](#)
 - [Templates](#)
 - [Variables](#)
 - [Filters](#)
 - [Tags](#)
 - [Comments](#)
 - [Template inheritance](#)

Getting Help

Language: en

Documentation version: 3.2

JEZYK SZABLONOW DJANGO

(DTL – DJANGO TEMPLATE LANGUAGE)

PRZYKŁAD SZABLONU DJANGO

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
    <a href="{{ story.get_absolute_url }}">
        {{ story.headline|upper }}
    </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

ELEMENTY SKLADNIOWE JĘZYKA SZABLONÓW DJANGO

- 1. Zmienne**
Zastępowane wartościami podczas generowania szablonu.
- 2. Tagi (znaczniki)**
Instrukcje obsługujące logikę szablonu, w tym instrukcje sterujące (warunki, pętle).
- 3. Filtry**
Funkcje modyfikujące wartości zmiennych.
- 4. Parametry filtra**
Dodatkowe parametry przyjmowane przez niektóre filtry, modyfikujące działanie tych filtrów
- 5. Notacja z kropką**
Odwołanie po kluczu, po atrybucie obiektu i po indeksie (dokładnie w tej kolejności) realizowane jest w ten sam sposób - za pomocą kropki.
- 6. Komentarze**

PC File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help intro - hello.html

intro > hello > hello_app > templates > hello_app > hello.html

Project DB Browser Structure Favorites

File Project settings.py hello\urls.py hello_app\urls.py views.py hello.html

__pycache__
init.py
asgi.py
settings.py
urls.py
wsgi.py
hello_app
migrations
templates
hello_app
hello.html
init.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
manage.py
venv library root

Terminal: Local +

Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.
(venv) C:\Users\jerem\PycharmProjects\intro>

1. ZMIENNE
{{ NAZWA_ZMIENNEJ }}

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Powitanie</title>
</head>
<body>
    Witaj, {{name}}!
</body>
</html>
```

ZMIENNE

- Nazwa zmiennej może być dowolną kombinacją znaków alfanumerycznych i podkreślnika (z wyłączeniem kombinacji, które zaczynają się od podkreślnika lub są liczbami)
- Jeżeli wprowadzona w szablonie zmienna nie istnieje, silnik języka szablonów Django w jej miejscu wstawi wartość zmiennej `string_if_invalid` (której domyślna wartość to pusty napis - ''). Odwołanie do nieistniejącej zmiennej NIE wywoła błędu!

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help intro - fruits.html

intro > hello > hello_app > templates > hello_app > fruits.html

Project DB Browser Structure Favorites

fr

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Owoce</title>
6 </head>
7 <body>
8
9 {% for fruit in fruits %}
10 {{ fruit }}
11 {% endfor %}
12
13 </body>
14 </html>

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

2. TAGI
{NAZWA_TAGA}%

TAGI (ZNACZNIKI)

- Niektóre znaczniki posiadające swój odpowiednik zamk傢acy i wymagaj  jego u ycia, np.:
`{% for ... %}{% endfor %}`
`{% if ... %}{% endif %}`
`{% block %}{% endblock %}`
- Django posiada oko o 20 wbudowanych znacznik w. Pe ne zestawienie wbudowanych znacznik w j zyka szablonów Django mo na znale c [tutaj](#)
- Django pozwala na definiowanie własnych tagów (patrz dodatki na koncu tej prezentacji).

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help intro - fruits.html

intro > hello > hello_app > templates > hello_app > fruits.html

Project DB Browser Structure Favorites

intro C:\Users\jerem\PycharmProjects\intro
hello
 hello
 __pycache__
 __init__.py
 asgi.py
 settings.py
 urls.py
 wsgi.py
 hello_app
 migrations
 templates
 hello_app
 fruits.html
 hello.html
 __init__.py
 admin.py
 apps.py
 models.py
 tests.py
 urls.py
 views.py
 manage.py
 venv library root

Terminal: Local +

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

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Owoce</title>
6 </head>
7 <body>
8 <p>{{ fruits|length }}</p>
9 </body>
10 </html>

3. FILTRY

{} NAZWA_ZMIENNEJ | NAZWA_FILTRA {}

TODO Problems

DB Execution Console

Terminal

Python Console

FILTRY

- Przykład filtra: filtr *lower* w działaniu na zmienną *name* - `{{ name | lower }}`
- Filtry można łańcuchować `{{ text | escape | linebreaks }}`
- *Django posiada około 60 wbudowanych filtrów. Pełne zestawienie wbudowanych filtrów języka szablonów Django można znaleźć [tutaj](#)*
- *Django pozwala na tworzenie własnych filtrów (patrz dodatki na końcu tej prezentacji)*

PC File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help intro - fruits.html

intro > hello > hello_app > templates > hello_app > fruits.html

Project DB Browser Structure Favorites

intro C:\Users\jerem\PycharmProjects\intro
hello
 hello
 pycache
 __init__.py
 asgi.py
 settings.py
 urls.py
 wsgi.py
 hello_app
 migrations
 templates
 hello_app
 fruits.html
 hello.html
 __init__.py
 admin.py
 apps.py
 models.py
 tests.py
 urls.py
 views.py
 manage.py
 venv library root

Terminal: Local +

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

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Owoce</title>
6 </head>
7 <body>
8 <p>{{ fruits|join:" . " }}</p>
9 </body>
10 </html>
11

settings.py x hello\urls.py x hello_app\urls.py x views.py x hello.html x

4. PARAMETRY FILTRA

{} NAZWA_ZMIENNEJ | NAZWA_FILTRA : PARAMETR {}

Project DB Browser Structure Favorites

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help intro - passengers.html

intro > hello > hello_app > templates > hello_app > passengers.html

Project DB Browser Structure Favorites

passengers.html

1 <!DOCTYPE html>

2 <html lang="en">

3 <head>

4 <meta charset="UTF-8">

5 <title>Pasażerowie</title>

6 </head>

7 <body>

8 <!-- Odwoływanie po indeksie -->

9 <p>{{ passengers.0 }}</p>

10 <!-- Odwoływanie po kluczu -->

11 <p>{{ passengers.0.age }}</p>

12 <!-- Odwoływanie po atrybutie -->

13 <p>{{ passengers.0.age.real }}</p>

14 </body>

15 ...

passengers.html

16 _init_.py

17 asgi.py

18 settings.py

19 urls.py

20 wsgi.py

21 hello_app

22 migrations

23 templates

24 hello_app

25 fruits.html

26 hello.html

27 passengers.html

28 __init__.py

29 admin.py

30 apps.py

31 models.py

32 tests.py

33 urls.py

34 views.py

35 manage.py

36 venv library root

37 External Libraries

38 Scratches and Consoles

Terminal: Local +

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

TODO Problems DB Execution Console Terminal Python Console

PyCharm 2020.3.5 available // Update... (39 minutes ago)

5. NOTACJA Z KROPKĄ

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

- Project:** intro (C:\Users\jerem\PycharmProjects\intro)
- File Structure:** The project contains a `hello` directory with `__init__.py`, `asgi.py`, `settings.py`, `urls.py`, and `wsgi.py`. It also contains a `hello_app` directory with `migrations`, `templates`, and `hello_app` sub-directories. The `fruits.html` file is selected in the `hello_app/templates/hello_app` folder.
- Code Editor:** The code editor displays the `fruits.html` template. A red box highlights the Jinja2 comment block:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Owoce</title>
    </head>
    <body>
        {% comment %}
            {{ Komentarz wielolinijkowy }}
        {% endcomment %}
    </body>
</html>
```
- Terminal:** The terminal shows the command prompt: `(venv) C:\Users\jerem\PycharmProjects\intro>`.
- Bottom Navigation:** Includes `TODO`, `Problems`, `DB Execution Console`, `Terminal`, and `Python Console`.
- Bottom Status:** `PyCharm 2020.3.5 available // Update... (10 minutes ago)`

6. KOMENTARZE

{# KOMENTARZ #}

KOMENTARZE

- *Zawartość komentarzy DTL nie jest traktowana przez silnik szablonów jako instrukcje DTL.*
- *Komentarze DTL w odróżnieniu od komentarzy HTML będą niewidoczne w ciele odpowiedzi. Silnik szablonów wycina wszystkie komentarze w trakcie generowania treści szablonu.*

PRZYKŁAD SZABLONU DJANGO (OMÓWIENIE)

```
{% extends "base_generic.html" %} Zmienna  
{% block title %}{{ section.title }}{% endblock %}  
  
{% block content %} Tag  
<h1>{{ section.title }}</h1>  
  
{% for story in story_list %} Notacja z kropką  
<h2>  
  <a href="{{ story.get_absolute_url }}">  
    {{ story.headline|upper }}  
  </a>  
</h2> Filtr  
<p>{{ story.tease|truncatewords:"100" }}</p>  
{% endfor %}  
{% endblock %} Parametr filtra
```

PRZYKŁADY ZASTOSOWAŃ

1. Treści statyczne
2. Warunki
3. Pętle
4. Dynamiczne linkowanie
5. Dziedziczenie
6. Kompozycja

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

- Project:** intro
- File Path:** intro > hello > hello_app > templates > hello_app > hello-again.html
- Code Editor:** Content of `hello-again.html`:

```
% load static
!DOCTYPE html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Powitanie</title>
    <link rel="stylesheet" href='{% static "hello_app/style.css" %}'>
    <script src='{% static "hello_app/script.js" %}'></script>
</head>
<body>
    <p>Witaj, świecie!</p>
    <img src='{% static "hello_app/hello.png" %}'>
</body>
</html>
```

- Structure View:** Shows the project structure with a red box highlighting the `static` folder under `hello_app`, which contains `hello.png`, `script.js`, and `style.css`.
- Terminal:** Shows the command prompt: `(venv) C:\Users\jerem\PycharmProjects\intro\hello>`
- Status Bar:** Shows the file encoding as `CRLF`, character set as `UTF-8`, and code style as `4 spaces`. It also indicates the Python version as `Python 3.8 (intro)`.

A red box highlights the `{% load static %}` directive in the code editor.

1. TRESCI STATYCZNE
ZNACZNIKI *LOAD, STATIC*

intro > hello > hello_app > templates > hello_app > hello-again.html

Add Configuration... ▶ ⌂

Project DB Browser

```
1  {% load static %}          ← static
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5    <meta charset="UTF-8">
6    <title>Powitanie</title>
7    <link rel="stylesheet" href='{% static "hello_app/style.css" %}'>
8    <script src='{% static "hello_app/script.js" %}'></script>
9  </head>
10 <body>
11   <p>Witaj, świecie!</p>
12   <img src='{% static "hello_app/hello.png" %}'>
13 </body>
14 </html>
```



1. TRESCI STATYCZNE
ZNACZNIKI *LOAD, STATIC*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Warunki</title>
</head>
<body>
    {% if age < 16 %}
        Osoba nieletnia
    {% elif age < 18 %}
        Osoba niepełnoletnia
    {% else %}
        Osoba pełnoletnia
    {% endif %}
</body>
</html>
```

2. WARUNKI

ZNACZNIKI IF, ELIF, ELSE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Pętle</title>
</head>
<body>
    {% for fruit in fruits %}
        {{ fruit }}
    {% endfor %}
</body>
</html>
```

3. PETLE

ZNACZNIK FOR

PC File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help intro - first_view.html

intro > hello > links > templates > links > first_view.html Add Configuration... ▶ ⚙ 🔍

Project DB Browser

urls.py

```
from django.urls import path

from links import views

urlpatterns = [
    path('first/', views.first, name='first'),
    path('second/', views.second, name='second'),
    path('third/<str:param>', views.third, name='third')
]
```

nazwy mapowań

first_view.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Title</title>
</head>
<body>
<a href="{% url 'third' 'adam' %}">
    Poprzednia strona
</a>
<a href="{% url 'second' %}">
    Następna strona
</a>
</body>
</html>
```

16:1 CRLF UTF-8 4 spaces Python 3.8 (intro)

4. DYNAMICZNE
LINKOWANIE
ZNACZNIK URL + NAZWY MAPOWAN

The screenshot shows the PyCharm IDE interface with two code editors open. On the left, the `urls.py` file is displayed:

```
from django.urls import path
from links import views

app_name = 'links'

urlpatterns = [
    path('first/', views.first, name='first'),
    path('second/', views.second, name='second'),
    path('third/<str:param>', views.third, name='third')
]
```

On the right, the `first_view.html` file is displayed:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <a href="{% url 'links:third' 'adam' %}">
        Poprzednia strona
    </a>
    <a href="{% url 'links:second' %}">
        Następna strona
    </a>
</body>
</html>
```

The `app_name` variable in `urls.py` and the URL patterns in `first_view.html` both reference the `'links'` application, which is highlighted in yellow in both files.

4. DYNAMICZNE LINKOWANIE PRZESTRZENI NAZW MAPOWAN

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

- layout.html** (left editor):
 - Contains the standard HTML structure: DOCTYPE, html, head, and body.
 - In the head section, there is a title block: `{% block title %}{% endblock %}`. A red box highlights this block, and a red arrow points from it to the corresponding block in `first_view.html`.
 - In the body section, there is a block: `{% block body %}{% endblock %}`. A red box highlights this block, and a red arrow points from it to the corresponding block in `first_view.html`.
- first_view.html** (right editor):
 - Starts with an extends statement: `{% extends "inheritance/layout.html" %}`.
 - Contains a title block: `{% block title %} Tytuł pierwszej strony {% endblock %}`. This block is highlighted with a red box.
 - Contains a body block: `{% block body %} Treść pierwszej strony. {% endblock %}`. This block is highlighted with a red box.

At the bottom right of the interface, there is a message in Polish: "Aktywuj system Windows" (Activate Windows system) with a sub-instruction: "Przejdz do ustawień, aby aktywować system Windows." (Go to settings to activate the Windows system).

Bottom navigation bar:

- TODO, Problems, DB Execution Console, Terminal, Python Console
- Event Log
- 10:1 CRLF UTF-8 4 spaces Python 3.8 (intro)

5. DZIEDZICZENIE

ZNACZNIKI *EXTENDS, BLOCK*

DZIEDZICZENIE

- Szablony dziedziczące nie muszą nadpisywać wszystkich bloków szablonu rodzica. Nienadpisane bloki dziedziczą zawartość po swoim rodzicu (puste pozostaną pustymi).
- Do zawartości bloku szablonu rodzica można odwoływać się za pomocą zmiennej `{{ block.super }}`. W ten sposób można łatwo rozszerzyć treść zawartą w bloku szablonu rodzica.
- Bloki mają płaską strukturę. Nie można tworzyć bloków o tej samej nazwie ani zagnieździć jedne bloki w innych.

The screenshot shows the PyCharm IDE interface with two code editors open. On the left, the Project tool window lists files: base.html, hello.html, and intro - base.html. The intro - base.html file is the current tab, showing a breadcrumb navigation path: intro > hello > inheritance > templates > inheritance > base.html. The base.html editor contains the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    {% include "inheritance/hello.html" %}
</body>
</html>
```

A red arrow points from the line '% include "inheritance/hello.html"' down to the hello.html editor on the right. The hello.html editor contains the following code:

```
<p>Witaj, świecie</p>
```

The PyCharm terminal at the bottom shows the project directory structure:

```
(venv) C:\Users\jerem\PycharmProjects\intro\hello>
(venv) C:\Users\jerem\PycharmProjects\intro\hello>
(venv) C:\Users\jerem\PycharmProjects\intro\hello>[REPL]
```

At the bottom of the interface, there are tabs for TODO, Problems, DB Execution Console, Terminal, Python Console, and Event Log. A status bar at the very bottom indicates: PyCharm 2020.3.5 available // Update... (13 minutes ago), 11:1 CRLF UTF-8 4 spaces, Python 3.8 (intro), and Event Log.

6. KOMPOZYCJA
ZNACZNIK **INCLUDE**

PYTANIA

PYTANIE I

W jaki sposób wyświetlony zostanie w przeglądarce napis {{ name }} znajdujący się w szablonie Django?

- A. name
- B. Wyświetlona zostanie wartość zmiennej pythonowej o nazwie name
- C. {{ name }}
- D. Żadne z powyższych

PYTANIE II

Czym jest forloop.counter?

- A. Metodą tworzącą instancję obiektu informującego czy pętla się skończyła
- B. Nie istnieje coś takiego
- C. To jest metoda która informuje ile było obieków pętli
- D. Zmienną występującą w pętli for, która informuje ile było obiegów pętli

PYTAÑIE III

Do czego używane jest element {# #} w szablonach?

- A. Do wprowadzania komentarzy
- B. Do wprowadzania logiki biznesowej
- C. Element nie jest obsługiwany, jego użycie podniesie wyjątek
- D. Żadne z powyższych

PYTANIE IV

Wskaż niepoprawny tag języka szablonów Django

- A. { % if % } { % else % }
- B. { % extends "base.html" % }
- C. { % while % }
- D. { % for key in dictionary % }

PYTANIE V

Jaki jest powód tworzenia zagnieżdzonej struktury folderów dla szablonów?

- A. Tworzy to przestrzeń nazw dla szablonów, zapewniając tym samym, że szablony które mają te same nazwy w kilku aplikacjach nie wywołają konfliktu.
- B. W ten sposób przekierowujem url na właściwy widok.
- C. Pozwala to na przekazanie danych do szablonu html.
- D. W ten sposób tworzymy urlconfig dla aplikacji.

PYTANIE VI

W jaki sposób używamy zmiennych w szablonach ?

- A. `{{ variable_name }}`
- B. `variable_name`
- C. `{ variable_name }`
- D. `{% variable_name %}`

PYTANIE VII

W jaki sposób poprawnie dodajemy url do strony z użyciem DTL (Django Template Language)?

- A. `Sample link`
- B. `Sample link`
- C. `Sample link`
- D. `Sample link`

PYTANIE VIII

Jaką nazwę przyjmuje parametr funkcji path() w miejscu
_____ celem wyrenderowania dynamicznego url w
szablonie ? path('/profile', views.index,
_____='dashboard')

- A. app_name
- B. template
- C. render
- D. name

PYTAÑIE IX

Która z poniższych funkcji umożliwia nam wysyłanie treści szablonu do klienta?

- A. `HttpResponseTemplate()`
- B. `render()`
- C. `HttpResponse()`
- D. `renderTemplate()`

PYTANIE X

Czym są pliki statyczne w Django?

- A. To pliki css, js itp.
- B. To inna nazwa dla widoków.
- C. To modele z bazy danych.
- D. To pliki ustawień, np. settings.py

PYTAŃIE XI

Które słowo jest używane jako tag służący do załadowywania innego szablonu wewnątrz szablonu ?

- A. include
- B. import
- C. from
- D. Żadne z powyższych

PYTANIE XII

Który znak używany jest do oznaczenia działania filtru na zmiennej ?

- A. &&
- B. &
- C. ||
- D. |

PYTAÑIE XIII

Którego znaku używamy, żeby zaznaczyć początek taga w szablonie ?

- A. %
- B. {
- C. {#
- D. {%

*MATERIALY
POMOCNICZE*

A screenshot of a web browser window showing the homepage of LearnDjango.com. The address bar displays the URL <https://learndjango.com>. The page header includes the site name "LearnDjango" and navigation links for "Tutorials", "Books", and "More". A search bar is also present. The main content area features the heading "Django Tutorials" and a subtitle "Up-to-date and covering the latest tools." Below this, logos for Django, Python, REST framework, Docker, Heroku, and PostgreSQL are displayed. A green button labeled "BROWSE ALL TUTORIALS" is centered below the logos.

Django Tutorials

Up-to-date and covering the latest tools.

django

 python™

django
REST
framework

 docker

 HEROKU



[BROWSE ALL TUTORIALS](#)

Featured Tutorials

[What is Django?](#)

[How to Learn Django \(2020\)](#)

<https://djangochat.com>

Django Chat
Django Testing and Debugging - Karen Tracey

Home Episodes Books Newsletter Twitter

Django Chat

A podcast on the Django Web Framework by William Vincent and Carlton Gibson.

Listen on Apple Podcasts [LISTEN & SUBSCRIBE](#)

Recent Episodes

53:20

1X 15 15

Django Chat

Django Testing and Debugging - Karen Tracey

RSS Twitter

Django

wsvincent/awesome-django: A curated list of awesome things related to Django

https://github.com/wsvincent/awesome-django

Search or jump to... / Pull requests Issues Marketplace Explore

wsvincent / awesome-django Public

Code Issues Pull requests 4 Discussions Actions Projects Wiki Security Insights

main 3 branches 0 tags Go to file Add file Code

wsvincent Merge pull request #153 from Rafat97/patch-1 ... b9f3320 4 days ago 452 commits

.github codeowners lint 4 months ago

_includes Adds semi-broken fork me ribbon 18 days ago

_sass/color_schemes Refactors theme and overall look. Fixes #133 5 months ago

assets Refactors theme and overall look. Fixes #133 5 months ago

.editorconfig Config updates 10 months ago

.gitignore Updates .gitignore to ignore .vendor 5 months ago

404.html A working GH Page 17 months ago

CNAME Update CNAME 17 months ago

Gemfile Move stuff around 17 months ago

LICENSE Creates LICENSE 2 years ago

README.md Merge pull request #153 from Rafat97/patch-1 4 days ago

About

A curated list of awesome things related to Django

awesomedjango.org

django awesome awesome-list

Readme CC0-1.0 License

5k stars 186 watching 824 forks

Releases

No releases published

Packages

Introduction · HonKit

Type to search

Need help? Talk to us!

Introduction

Installation

Installation (chromebook)

How the Internet works

Introduction to command line

Python installation

Code editor

Introduction to Python

What is Django?

Django installation

Your first Django project!

Django models

Django admin

 Support our work and donate to our project! 

Donate now!

Hide Outline Language Font Settings

Django Girls Tutorial

chat on gitter

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
To view a copy of this license, visit <https://creativecommons.org/licenses/by-sa/4.0/>

Welcome

Welcome to the Django Girls Tutorial! We are happy to see you here. :) In this tutorial, we will take you on a journey under the hood of web technologies, offering you a glimpse of all the bits and pieces that need to come together to make the web work as we know it.

As with all unknown things, this is going to be an adventure - but no worries, since you already worked up the courage to be here, you'll be just fine. :)

Introduction

Have you ever felt that the world is more and more about technology to which you cannot (yet) relate? Have you ever wondered how to create a website but have never had enough motivation to start? Have you ever thought that the software world is too complicated for you to even try doing something on your own?

Well, we have good news for you! Programming is not as hard as it seems and we want to show you how fun it can be.

This tutorial will not magically turn you into a programmer. If you want to be good at it, you need months or even years of learning and practice. But we want to show you that programming or creating websites is

> Next

Matt Layman

https://www.mattlayman.com

MATT LAYMAN

My Gear Most Popular Articles

Twitter GitHub YouTube Twitch LinkedIn



I'm Matt Layman.
I want to help you learn more about
Django and **Python**.



Django Riffs

Learn Django on the go!



Understand Django

My mega article series to help you learn Django



Programista Back-End
Stabilne zatrudnienie w topowych firmach IRKL

Otwórz

Adam Johnson

[Home](#) | [Blog](#) | [Books](#) | [Projects](#) | [Colophon](#) | [Contact](#)

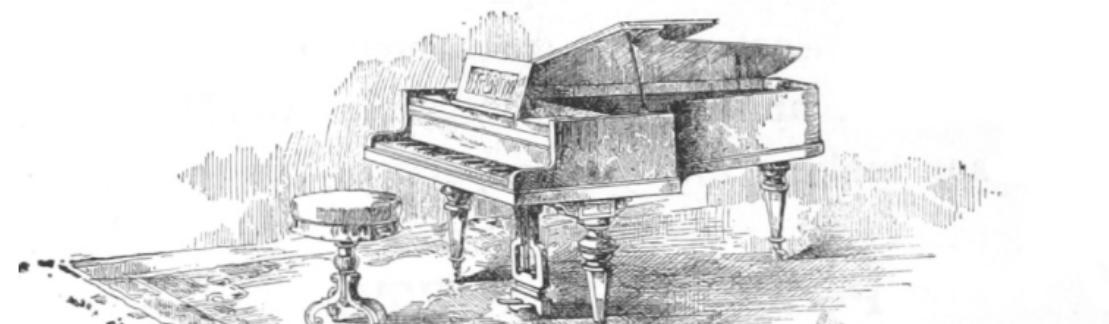
Finding the Best Posts

See my [top posts page](#) for a selection of my best posts. Or search the site with **Duck Duck Go**:

Recent Posts

Announcing WhiteNoise 6

2022-02-09



CodingEntrepreneurs - YouTube +

https://www.youtube.com/c/CodingEntrepreneurs

YouTube PL

Szukaj

Główna

Odkrywaj

Subskrypcje

Biblioteka

Historia

Twoje filmy

Do obejrzenia

e

Pokaż więcej

SUBSKRYPCJE

Kanał Sportowy

Sigur Rós

Clear Code

freeCodeCamp.org

Just Join IT

Automation Step b...

Copernicus Center ...

Learn to Code

Build Projects from Scratch. Step-by-Step

Project Tutorials

CodingEntrepreneurs 195 tys. subskrybentów

SUBSKRYBUJESZ

GŁÓWNA WIDEO PLAYLISTY SPOŁECZNOŚĆ KANAŁY INFORMACJE

Try Django 3.2 - Python Web Development Tutorial Series ► ODTWÓRZ WSZYSTKIE

Try Django 3.2 is a series to teach you the fundamentals of creating web applications with Python & Django by building a real project step-by-step. We partnered with DigitalOcean to make this seri...

Part 1: 1 - Welcome to Try Django 3.2 - Python & Django 3.2...
CodingEntrepreneurs 47 tys. wyświetleń • 5 miesięcy temu

Part 2: 2 - Demo after 73 Parts - Python & Django 3.2 Tutoria...
CodingEntrepreneurs 19 tys. wyświetleń • 5 miesięcy temu

Part 3: 3 - Requirements - Python & Django 3.2 Tutorial Series
CodingEntrepreneurs 9,4 tys. wyświetleń • 5 miesięcy temu

Part 4: 4 - Python & Python Virtual Environment Setup on...
CodingEntrepreneurs 8,1 tys. wyświetleń • 5 miesięcy temu

Part 5: 5 - Python & Python Virtual Environments Setup on...
CodingEntrepreneurs 6,5 tys. wyświetleń • 5 miesięcy temu

Part 6: 6 - Setup a Django Project - Python & Django 3.2 Tutoria...
CodingEntrepreneurs 8,2 tys. wyświetleń • 5 miesięcy temu

YouTube PL

JustDjango - YouTube

https://www.youtube.com/c/JustDjango

Szukaj

JD

JustDjango

Become a professional Django developer

Learn Django Professionally

Główna

Odkrywaj

Subskrypcje

Biblioteka

Historia

Twoje filmy

Do obejrzenia

e

Pokaż więcej

JustDjango

68,3 tys. subskrybentów

SUBSKRYBUJESZ

GŁÓWNA

VIDEO

PLAYLISTY

SPOŁECZNOŚĆ

KANAŁY

INFORMACJE

Przesłane filmy

ODTWÓRZ WSZYSTKIE

Dynamic Forms with Django and Htmx

Django Dynamic Forms Tutorial with Htmx

17 tys. wyświetleń • 5 miesięcy temu

PayPal payments with Django and React

PayPal Payments Tutorial with Django and React

5 tys. wyświetleń • 6 miesięcy temu

Dockerize Django in 5 minutes

How to Dockerize Django in 5 minutes

9,4 tys. wyświetleń • 6 miesięcy temu

stripe

Payments tutorial with Django

1:10:27

Is Django becoming obsolete?

R.I.P.

What is the future of Django / Is Django becoming... 59 tys. wyświetleń • 1 rok temu

Salary data for Django developers

Introducing JustDjango Salaries: Salary data for...

4:03

Popularne filmy

ODTWÓRZ WSZYSTKIE

The Dumbfounds - YouTube

https://www.youtube.com/channel/UC33uwXXDrI5TxG4IXnjS28g

YouTube PL

Szukaj

Główna

Odkrywaj

Subskrypcje

Biblioteka

Historia

Twoje filmy

Do obejrzenia

e

Pokaż więcej

SUBSKRYBUJESZ

The Dumbfounds

12 tys. subskrybentów

GŁÓWNA WIDEO PLAYLISTY SPOŁECZNOŚĆ KANAŁY INFORMACJE

Django Fundamentals For Beginners - Get Started Quickly! ► ODTWÓRZ WSZYSTKIE

Django 2 Setup Tutorial 4:57 Django 2 Fundamentals 0:33 Django 2 URLs Tutorial 10:43 Django 2 Templates Tutorial 10:49 Django 2 Models Tutorial 9:16 Django 2 CBVs Tutorial 5:56

Django 2 Setup Tutorial For Beginners (2018) Django 2 Fundamentals Tutorial For Beginners -... Django 2 URLs Tutorial For Beginners (2018) Django Templates Tutorial For Beginners (2018) Django 2 Models Tutorial For Beginners (2018) Django 2 Class-Based Views Tutorial (+ Generic Views)

The Dumbfounds 8,6 tys. wyświetleń • 3 lata temu The Dumbfounds 4,5 tys. wyświetleń • 3 lata temu The Dumbfounds 22 tys. wyświetleń • 3 lata temu The Dumbfounds 10 tys. wyświetleń • 3 lata temu The Dumbfounds 5,1 tys. wyświetleń • 3 lata temu The Dumbfounds 8,3 tys. wyświetleń • 3 lata temu

Django 2 Tutorial: Build A Budget Application ► ODTWÓRZ WSZYSTKIE

contact@thedumbfounds.com

Twitter

Pyplane - YouTube

https://www.youtube.com/c/Pyplane

Szukaj

YouTube PL

Główna

Odkrywaj

Subskrypcje

Biblioteka

Historia

Twoje filmy

Do obejrzenia

e

Pokaż więcej

SUBSKRYPCJE

Kanał Sportowy

Sigur Rós

Clear Code

freeCodeCamp.org

Just Join IT

Automation Step b...

Copernicus Center ...

Pyplane

22,5 tys. subskrybentów

SUBSKRYBUJESZ

GŁÓWNA

VIDEO

PLAYLISTY

SPOŁECZNOŚĆ

KANAŁY

INFORMACJE

Python Django tutorial 2021 - full course from beginner to advanced [preview]

Pyplane • 3,5 tys. wyświetleń • 10 miesięcy temu

Coming up next week in one single part - django full course. We will cover among others: - django concepts (models, views, templates, signals and more!) - pandas dataframes - matplotlib and...

DJANGO FULL COURSE PREVIEW

3:04

Popularne filmy

ODTWÓRZ WSZYSTKIE

DJANGO 3 FULL COURSE

django

7:00:15

django

part 1

16:05

+django

20:49

django

PART 1

26:48

django

PDF +

34:16

django

5:49:45

DODATKI

WLASNE FILTRY

pythonProject3 > example

Project

DB Browser

pythonProject3 C:\Users\jerem\PycharmProjects\pythonProject3

example

- _pycache_
- migrations
- templates
- templatetags
 - __pycache__
 - __init__.py
 - custom_tags.py
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
- new
- venv library root
 - db.sqlite3
 - manage.py

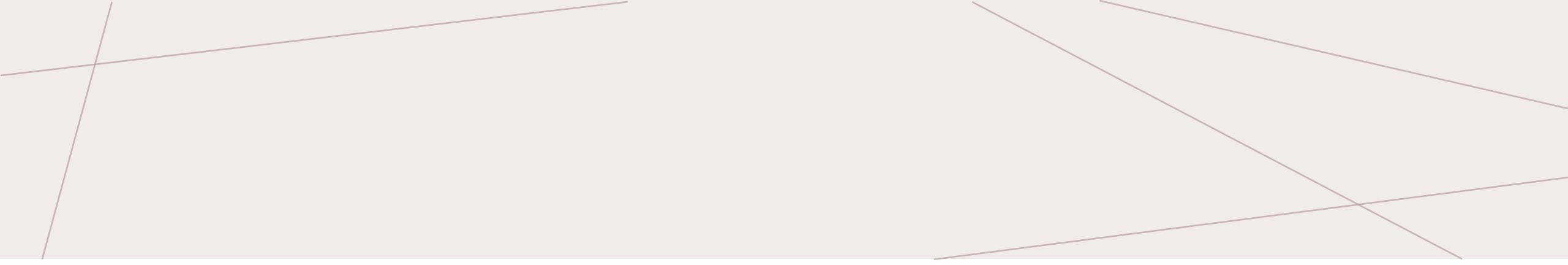
External libraries

Terminal: Local +

```
graph TD; pythonProject3[pythonProject3] --> example[example]; example --> _pycache_[__pycache__]; example --> migrations[migrations]; example --> templates[templates]; example --> templatetags[templatetags]; templatetags --> __pycache__1[__pycache__]; templatetags --> __init__1['__init__.py']; templatetags --> custom_tags[custom_tags.py]; templatetags --> __init__2['__init__.py']; templatetags --> admin[admin.py]; templatetags --> apps[apps.py]; templatetags --> models[models.py]; templatetags --> tests[tests.py]; templatetags --> urls[urls.py]; templatetags --> views[views.py]; example --> new[new]; example --> venv[venv library root]; venv --> db[db.sqlite3]; venv --> manage[manage.py];
```

Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.
(venv) C:\Users\jerem\PycharmProjects\pythonProject3>

WLASNE FILTRY
BIBLIOTEKA TAGOW



*Własne tagi i filtry umieszczamy w odrębnych plikach (modułach pythona) nazywanych **bibliotekami tagów**. Biblioteki tagów powinny znajdować się wewnątrz aplikacji, w folderze **templatetags**. Folder templatetags powinien być jednocześnie pakietem pythona (pamiętaj o dodaniu pustego pliku `__init__.py`).*

Aby Django traktował moduł pythona jak bibliotekę tagów należy zdefiniować w niej zmienna `register`, której wartością jest instancja obiektu `django.template.Library()`

pythonProject3 > example > templatetags > custom_tags.py

Project DB Browser

pythonProject3 C:\Users\jerem\PycharmProjects\pythonProject3
example
pycache
migrations
templates
templatetags
pycache
__init__.py
custom_tags.py
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
new
venv library root
db.sqlite3
manage.py

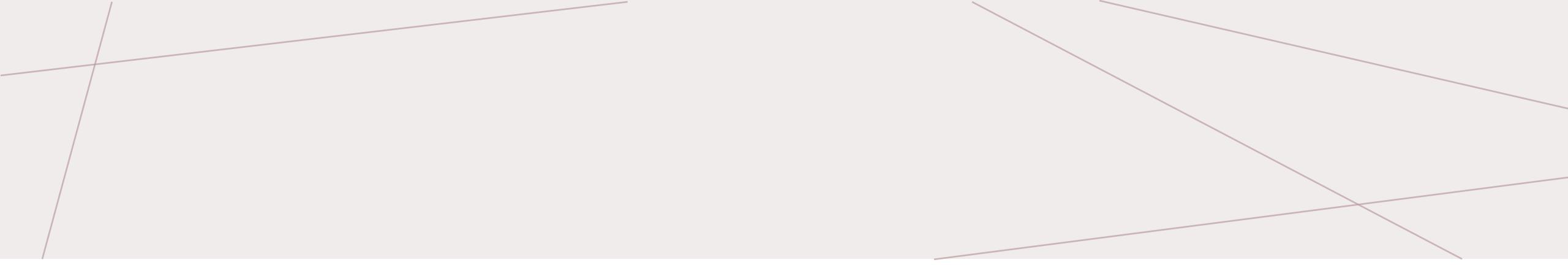
External libraries

Terminal: Local +

```
from django import template
register = template.Library()
```

Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.
(venv) C:\Users\jerem\PycharmProjects\pythonProject3>

WLASNE FILTRY
BIBLIOTEKA TAGOW



W bibliotece tagów definiujemy oraz rejestrujemy własne tagi oraz filtry.

pythonProject3 > example > templatetags > custom_tags.py

Project

DB Browser

Structure

Favorites

```
from django import template  
  
register = template.Library()  
  
def trim(value):  
    if len(value) > 6:  
        return value[:6]  
    return value
```

pythonProject3 C:\Users\jerem\PycharmProjects\pythonProject3
example
 > __pycache__
 > migrations
 > templates
 > templatetags
 > __pycache__
 > __init__.py
 > custom_tags.py
 > __init__.py
 > admin.py
 > apps.py
 > models.py
 > tests.py
 > urls.py
 > views.py
 > new
 > venv library root
 db.sqlite3
 manage.py

External Libraries

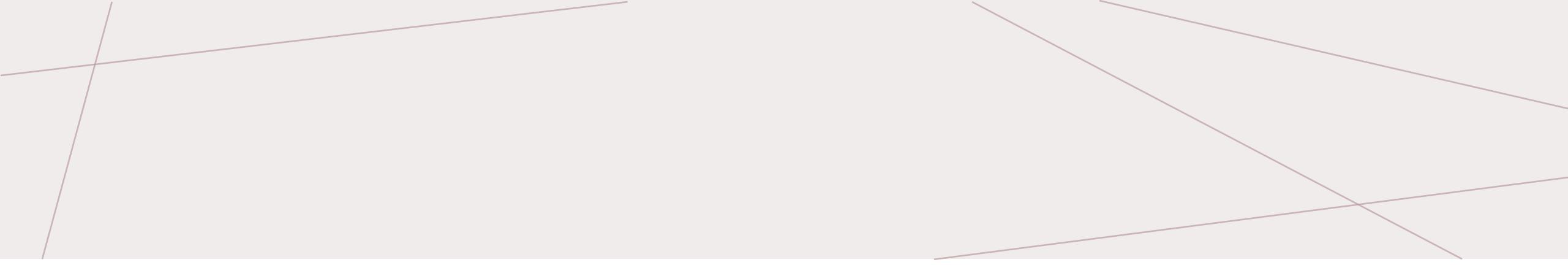
Terminal: Local +

Microsoft Windows [Version 10.0.19042.1348]

(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

(venv) C:\Users\jerem\PycharmProjects\pythonProject3>

WŁASNE FILTRY
DEFINIOWANIE FILTRA



Instnieją dwa sposoby rejestrowania filtra:

1. *metoda filter obiektu klasy django.template.Library*
2. *dekorator django.template.Library.filter*

pythonProject3 > example > templatetags > custom_tags.py

Project

DB Browser

Custom

Project

```
from django import template

register = template.Library()

def trim(value):
    if len(value) > 6:
        return value[:6]
    return value

register.filter('trim', trim)
```

> new
> venv library root
 db.sqlite3
 manage.py

> External Libraries

Terminal: Local +

Microsoft Windows [Version 10.0.19042.1348]

(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

(venv) C:\Users\jerem\PycharmProjects\pythonProject3>

WLASNE FILTRY
REJESTRACJA FILTRA I

```
from django import template

register = template.Library()

@register.filter
def trim(value):
    if len(value) > 6:
        return value[:6]
    return value
```

WŁASNE FILTRY
REJESTRACJA FILTRA II

pythonProject3 > example > views.py

custom_tags.py

```
from django import template

register = template.Library()

@register.filter
def trim(value):
    if len(value) > 6:
        return value[:6]
    return value
```

views.py

```
from django.shortcuts import render

def index(request):
    return render(
        request,
        'example/index.html',
        context={
            "x": 'bardzo dlugi',
            'y': 'abc',
        }
    )
```

index.html

```
{% load custom_tags %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>Wartość: <b>{{ x }}</b> po zastosowaniu filtra trim ma wartość: <b>{{ x | trim }}</b></p>
    <p>Wartość: <b>{{ y }}</b> po zastosowaniu filtra trim ma wartość: <b>{{ y | trim }}</b></p>
</body>
</html>
```

WŁASNE FILTRY
UŻYCIE FILTRA

custom_tags.py

```
from django import template

register = template.Library()

@register.filter
def trim(value):
    if len(value) > 6:
        return value[:6]
    return value
```

views.py

```
from django.shortcuts import render

def index(request):
    return render(
        request,
        'example/index.html',
)
```

index.html

```
{% load custom_t
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>Wartość: <b>{{ x }}</b> po zastosowaniu filtra trim ma wartość: <b>{{ x | trim }}</b></p>
    <p>Wartość: <b>{{ y }}</b> po zastosowaniu filtra trim ma wartość: <b>{{ y | trim }}</b></p>
</body>
</html>
```

127.0.0.1:8000/example/

Wartość: **bardzo dlugi napis** po zastosowaniu filtra trim ma wartość: **bardzo**

Wartość: **abc** po zastosowaniu filtra trim ma wartość: **abc**

WŁASNE FILTRY
UŻYCIE FILTRA

WLASNE TAGI

WŁASNE TAGI

Tagi mają bardziej złożoną strukturę niż filtry. Filtr to po prostu funkcja do której przekazujemy jakiś parametr/parametry. W wyniku działania tej funkcji otrzymujemy przetworzoną wartość przekazanego/ych parametru/ów.

Tagi mogą realizować znaczne bardziej skomplikowane operacje. Poznaliśmy tag, który przyjmuje nazwę mapowania i zwraca nam odpowiadający tej nazwie endpoint (`{% url %}`). W przypadku tego taga podobny efekt można byłoby łatwo osiągnąć za pomocą filtrów. Ale poznaliśmy też tag, który realizuje pętle (`{% for ... %}`), czy warunek (`{% if ... %}`). Tego typu tagi mają bardziej złożoną strukturę. Dlatego istnieje kilka sposobów tworzenia własnych tagów. W ramach prezentacji omówimy tylko najprostszy ze sposobów.

WŁASNE TAGI

*Najprostszy ze sposobów tworzenia własnego taga polega na zamianie funkcji/dekoratora filter (wykorzystywanego przy definiowaniu własnych filtrów) na funkcje/dekorator **simple_tag**. Reszta jest identyczna jak w przypadku definiowania własnych filtrów. Ta metoda jest wystarczająca dla prostszych tagów. Niestety nie wszystkie tagi można w ten sposób utworzyć. Jeżeli tag, który chcemy utworzyć koduje bardziej złożoną logikę będziemy potrzebowali innych metod tworzenia.*

Szczegółowe omówienie, obejmujące wszystkie możliwe metody tworzenia własnych tagów można znaleźć w [dokumentacji Django](#).

pythonProject3 > example > templatetags > custom_tags.py

Project

DB Browser

Structure

Favorites

custom_tags.py

```
import datetime
from django import template

register = template.Library()

def current_time(format_string):
    return datetime.datetime.now().strftime(format_string)
```

```
register.simple_tag(current_time)
```

Project

- pythonProject3 C:\Users\jerem\PycharmProjects\pythonProject3
 - example
 - _pycache_
 - migrations
 - templates
 - templatetags
 - _pycache_
 - __init__.py
 - custom_tags.py
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - new
 - venv library root
 - db.sqlite3
 - manage.py
- External Libraries

Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\pythonProject3>

TODO

Problems

DB Execution Console

Terminal

Python Console

WLASNE TAGI
REJESTRACJA TAGA I

pythonProject3 > example > templatetags > custom_tags.py

Project

DB Browser

Structure

Favorites

custom_tags.py

```
import datetime
from django import template

register = template.Library()

@register.simple_tag
def current_time(format_string):
    return datetime.datetime.now().strftime(format_string)
```

Project

- pythonProject3 C:\Users\jerem\PycharmProjects\pythonProject3
 - example
 - _pycache_
 - migrations
 - templates
 - templatetags
 - _pycache_
 - __init__.py
 - custom_tags.py
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - new
 - venv library root
 - db.sqlite3
 - manage.py
- External Libraries

Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\pythonProject3>[]

TODO

Problems

DB Execution Console

Terminal

Python Console

PEP 8: W391 blank line at end of file

WLASNE TAGI
REJESTRACJA TAGA II

The screenshot shows the PyCharm IDE interface with a Django project structure. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, DB Navigator, and Help. The current file is index.html.

Project pane:

- pythonProject3
- example
- templates
- example
- index.html

DB Browser pane:

Structure pane:

Favorites pane:

Code Editor panes:

- custom_tags.py**:

```
import datetime
from django import template

register = template.Library()

@register.simple_tag
def current_time(format_string):
    return datetime.datetime.now().strftime(format_string)
```
- views.py**:

```
from django.shortcuts import render

def index(request):
    return render(
        request,
        'example/index.html',
        context={
            "x": 'bardzo dlugi napis',
            'y': 'abc',
        }
    )
```
- index.html**:

```
{% load custom_tags %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    {% current_time "%Y-%m-%d %H:%M" %}
</body>
</html>
```

Bottom navigation bar: TODO, Problems, DB Execution Console, Terminal, Python Console.

WLASNE TAGI
UZYCIE TAGA

The screenshot shows a PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, DB Navigator, Help.
- Project:** pythonProject3 - index.html
- Custom Tags (custom_tags.py):**

```
import datetime
from django import template

register = template.Library()

@register.simple_tag
def current_time(format_):
    return datetime.datetime.now().strftime(format_)
```
- Views (views.py):**

```
from django.shortcuts import render
```
- index.html:**

```
{% load custom %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    {{ current_time "%Y-%m-%d %H:%M" }}
</body>
</html>
```
- Browser Preview:** A modal window displays the output of the custom tag: **2021-11-24 16:02**.
- Bottom Navigation:** TODO, Problems, DB Execution Console, Terminal, Python Console.

WLASNE TAGI
UZYCIE TAGA

PODSUMOWANIE

*Własne tagi i filtry umieszczamy w odrębnych plikach (modułach pythona). Te pliki powinny znajdować się wewnątrz aplikacji, w folderze **templatetags**. Folder templatetags powinien być jednocześnie pakietem pythona (pamiętaj o dodaniu pustego pliku `__init__.py`).*



FRONTEND

WEB DESIGN

PROJEKTOWANIE RESPONSYWNE

FRAMEWORKI CSS

Do najpopularniejszych framework-ów css należą:

1. Bootstrap
2. Bulma
3. Materialize CSS
4. Foundation
5. Skeleton
6. Tailwind CSS

Bootstrap - The most popular HT x

← → 🔍 https://getbootstrap.com

Zaloguj

Home Docs Examples Icons Themes Blog

Download

Build fast, responsive sites with Bootstrap

Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.



Bulma: Free, open source, and modern CSS framework for building responsive, mobile first projects on the web.

BULMA

Documentation Videos Expo Love Backers ... GitHub Twitter Download Sponsor Search

Bulma: the modern CSS framework that just works.

Bulma is a free, open source framework that provides ready-to-use frontend components that you can easily combine to build responsive web interfaces.

No CSS knowledge required.

Star 44,657 downloads 832k/month

npm install bulma



Documentation - Materialize x +

[materializecss.com](#)

The screenshot shows the Materialize CSS documentation website. The header features the Materialize logo (a stylized red 'M') and the text '1.0.0'. A search bar with a magnifying glass icon is positioned above a sidebar menu. The sidebar includes links for 'About', 'Getting Started', 'CSS', 'Components', 'JavaScript', 'Forms', 'Mobile', 'Showcase', and 'Themes'. The main content area has a large red 'Materialize' title and a subtitle 'A modern responsive front-end framework based on Material Design'. Below this are two red buttons: 'GET STARTED' and 'UPGRADE FROM 0.100.2'. A note 'Release: 1.0.0' is visible. A Testim advertisement is present, showing three cartoon robots and the text 'Reduce your bug escape rate with Testim's AI-powered test automation platform. Try it for free.' A note at the bottom right says 'Aktywuj system Windows' and 'Przejdz do ustawień, aby aktywować system Windows.'

Materialize

A modern responsive front-end framework
based on Material Design

GET STARTED UPGRADE FROM 0.100.2

Release: 1.0.0

testim

Reduce your bug escape rate with Testim's AI-powered test automation platform. Try it for free.

ads via Carbon

Aktywuj system Windows
Przejdz do ustawień, aby aktywować system Windows.

The most advanced responsive fi X +

← → C 🔒 get.foundation

Foundation Showcase Develop Tutorials Get Involved Docs Getting Started

Foundation

The most advanced responsive front-end framework in the world.

[Download Foundation 6](#)

★ 29.2k GitHub stars
🐦 @FoundationCSS

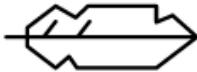
A large, stylized illustration of an astronaut's head and shoulders. The astronaut wears a white space suit with a grey helmet and an orange visor. They have glasses and a small smile. The background features a dark blue space with numerous small white stars and a single, bright yellow-orange comet streaking across the upper left. In the lower foreground, there are dark, silhouetted mountain peaks under a hazy orange glow, suggesting either a sunset or sunrise on another planet.

S Skeleton: Responsive CSS Boilerplate +

Niezabezpieczona | getskeleton.com

A dead simple, responsive boilerplate.

[DOWNLOAD](#)



Light as a feather at ~400 lines & built with mobile in mind.



Styles designed to be a starting point, not a UI framework.



Quick to start with zero compiling or installing necessary.

Aktywuj system Windows
Przejdz do ustawieñ, aby aktywowaæ system Windows.

INTRO CODE EXAMPLES MORE

Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.

A utility-first CSS framework packed with classes like `flex`, `pt-4`, `text-center` and `rotate-90` that can be composed to build any design, directly in your markup.

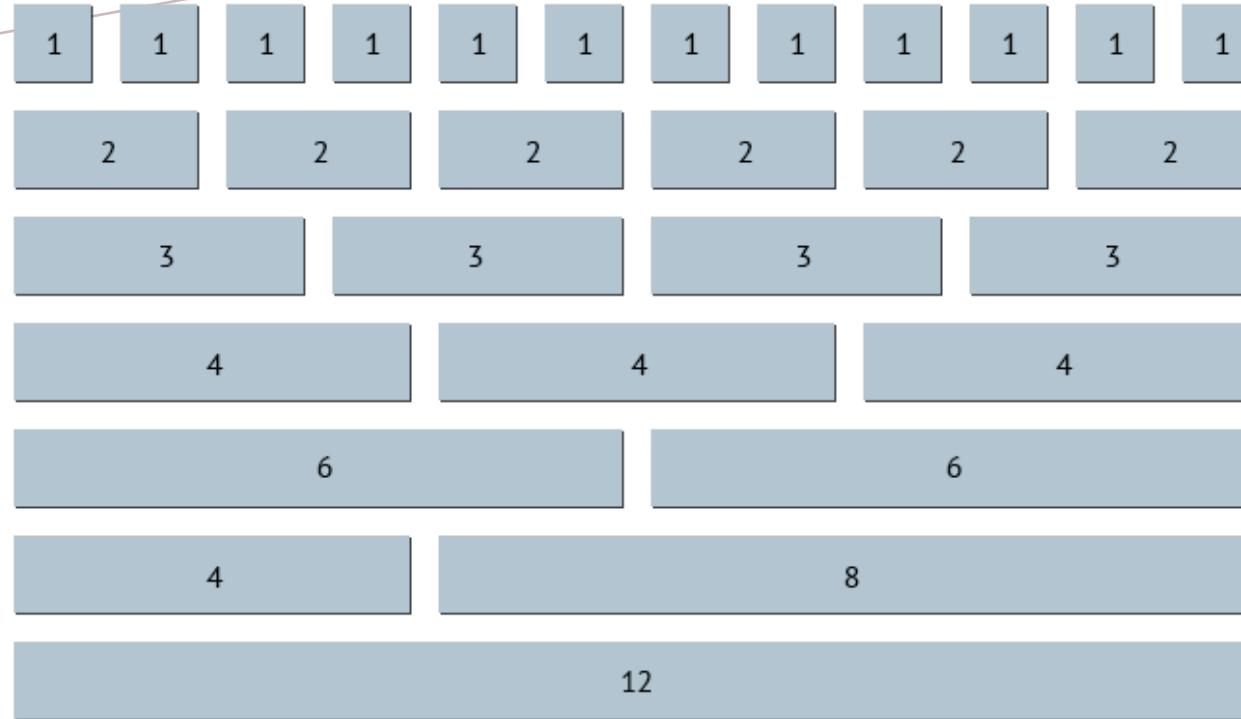
[Get started](#)

`$ npm install tailwindcss`

The image shows a screenshot of a web browser displaying the Tailwind CSS website at tailwindcss.com. The page features a large, bold headline: "Rapidly build modern websites without ever leaving your HTML.". Below the headline, there is a descriptive paragraph about the utility-first nature of the framework. At the bottom left, there is a "Get started" button and a terminal-like input field containing the command "\$ npm install tailwindcss". On the right side, there is a dark callout box containing a snippet of HTML and CSS code. The code includes a figure element with an image and a blockquote with a large text paragraph. The overall design is clean and modern, utilizing the Tailwind CSS classes mentioned in the text.

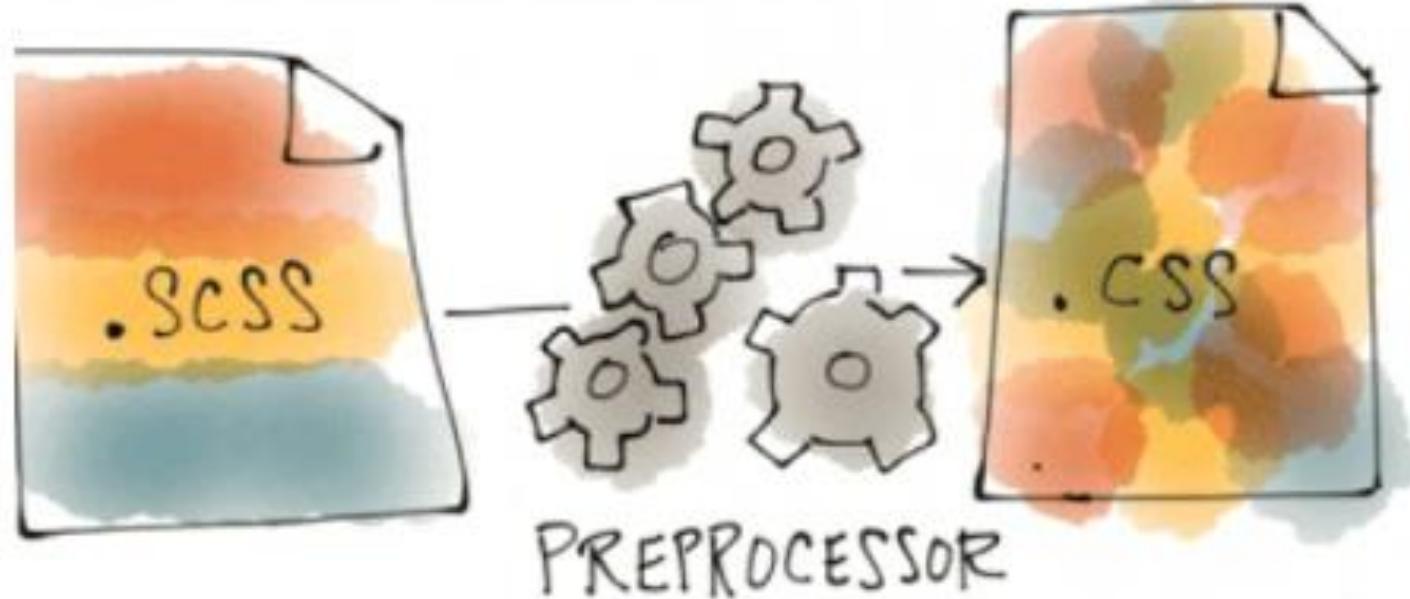
```
1 <figure class="bg-gray-100 rounded-xl p-8">
2   
3   <div class="pt-6 space-y-4">
4     <blockquote>
5       <p class="text-lg font-semibold">
6         Tailwind CSS is the only framework that I've seen scale
```

BOOTSTRAP



SIATKA BOOTSTRAP

PREPROCESORY CSS



PREPROCESSING CSS

Do najpopularniejszych preprocessor-ów css należą:

1. SASS (**Syntactically Awesome Style Sheets**)
2. LESS (**LEaner Style Sheets**)
3. Stylus
4. PostCSS

Sass: Syntactically Awesome Style x +

← → ⌂ https://sass-lang.com ⌂ Zaloguj ...

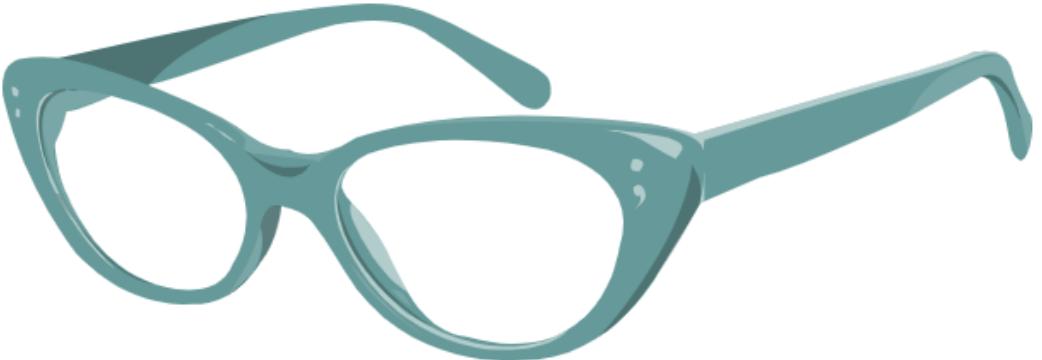
Sass

Install Learn Sass Blog Documentation Get Involved

Search CTRL K

CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.



Current Releases: [Dart Sass 1.43.5](#) [LibSass 3.6.5](#) [Ruby Sass](#)  [Implementation Guide](#)

A screenshot of a web browser displaying the 'Getting started | Less.js' page from lesscss.org. The page has a dark blue header with navigation links for Overview, Using Less.js, Functions, In-Depth Guide, Tools, and GitHub. The main content features a large white 'less' logo on a blue background, followed by the tagline 'It's CSS, with just a little more.' Below this, there are sections for 'Use with Node.js:' showing command-line examples, and 'Or the browser:' showing HTML code for linking to a LESS file or using a CDN script.

less Getting started | Less.js

Overview Using Less.js Functions In-Depth Guide Tools GitHub

lesscss.org

{less}

It's CSS, with just a little more.

Use with Node.js:

```
npm install -g less  
> lessc styles.less styles.css
```

Or the browser:

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />  
<script src="https://cdn.jsdelivr.net/npm/less@4.1.1" ></script>
```

The screenshot shows a web browser window displaying the Stylus language website at <https://stylus-lang.com>. The page features a large Stylus logo and the tagline "EXPRESSIVE, DYNAMIC, ROBUST CSS". Below this, two code snippets are shown in red font:

```
# CSS needs a hero
body {
  font: 12px Helvetica, Arial, sans-serif;
}
a.button {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
```

```
# What if we could omit braces?
body
  font: 12px Helvetica, Arial, sans-serif;
a.button
  -webkit-border-radius: 5px;
```

To the right of the code snippets is a vertical sidebar with a red background containing various Stylus features listed in white text:

- TRY STYLUS ONLINE!
- SELECTORS
- VARIABLES
- INTERPOLATION
- OPERATORS
- MIXINS
- FUNCTIONS
- KEYWORD ARGUMENTS
- BUILT-IN FUNCTIONS
- REST PARAMS
- COMMENTS



Increase code readability

Add vendor prefixes to CSS rules using values from Can I Use. [Autoprefixer](#) will use the data based on current browser popularity and property support to apply prefixes for you.

```
:fullscreen {  
}  
  
CSS input
```

```
:webkit-full-screen {  
}  
:-ms-fullscreen {  
}  
:fullscreen {  
}
```



```
CSS output
```

OPEN CHAT



SERWER



Klient

*ARCHITEKTURA
Klient-Serwer*