

Zaawansowane Aplikacje Internetowe

► Framework Django

- Podstawowa aplikacja
- Szablony i Moduły
- Dostęp do bazy danych
- Wdrożenie projektu



- **Możliwości Django**
- **Moduły, Frameworki i PyRozwiązania**
- **Instalacja i Uruchomienie**
- **Katalogi oraz PyPodejście do Web-aplikacji**
- **Projekty i Aplikacje**

Django – the web framework

Możliwości



- **zbudowany na architekturze MTV (*model-template-view*)**, pokrewnej do architektury MVC (*model-view-controller*)
- **framework Webowy**
ułatwiający tworzenie złożonych witryn, silnie opartych o bazy danych
- **rozszerzalność:**
stosowanie modułowości i komponentów (*reusability & „pluggability”*)
- **efektywność:**
mniej kodu, szybszy rozwój (*rapid development*), zasada DRY

Komponenty frameworka

Możliwości



- **Web serwer** – lekki i samodzielny serwer **deweloperski** i testowy
- system translacji, **walidacji** oraz **serializacji** danych FORM<=>DB
- system **szablonów HTML** stosujący koncepcję dziedziczenia z OOP
- elastyczny framework **cache**'ujący umożliwiający rozszerzanie
- system **językowy i18n** oraz przetłumaczone moduły Django
- system **serializacji** danych z/do: XML, JSON, modelu Django
- interfejs do wbudowanego w Python frameworka do Unit-testów

Wbudowane w Django aplikacje

Możliwości



- elastyczny system **autentykacji**
- dynamicznie generowany **interfejs administracyjny** CRUD
- narzędzia do generowania **kanałów dystrybucji** treści RSS oraz ATOM
- moduł **SITES** obsługujący wiele Witryn i Aplikacji na 1 instancji Django
- narzędzie do generowania **mapy witryny** (*plik XML Sitemap*)
- wbudowane algorytmy **łagodzenia ataków** CSRF, CSS, SQL-injection, password-cracking oraz innych popularnych
- framework do tworzenia aplikacji GIS

Współpraca Django z serwerami

Możliwości



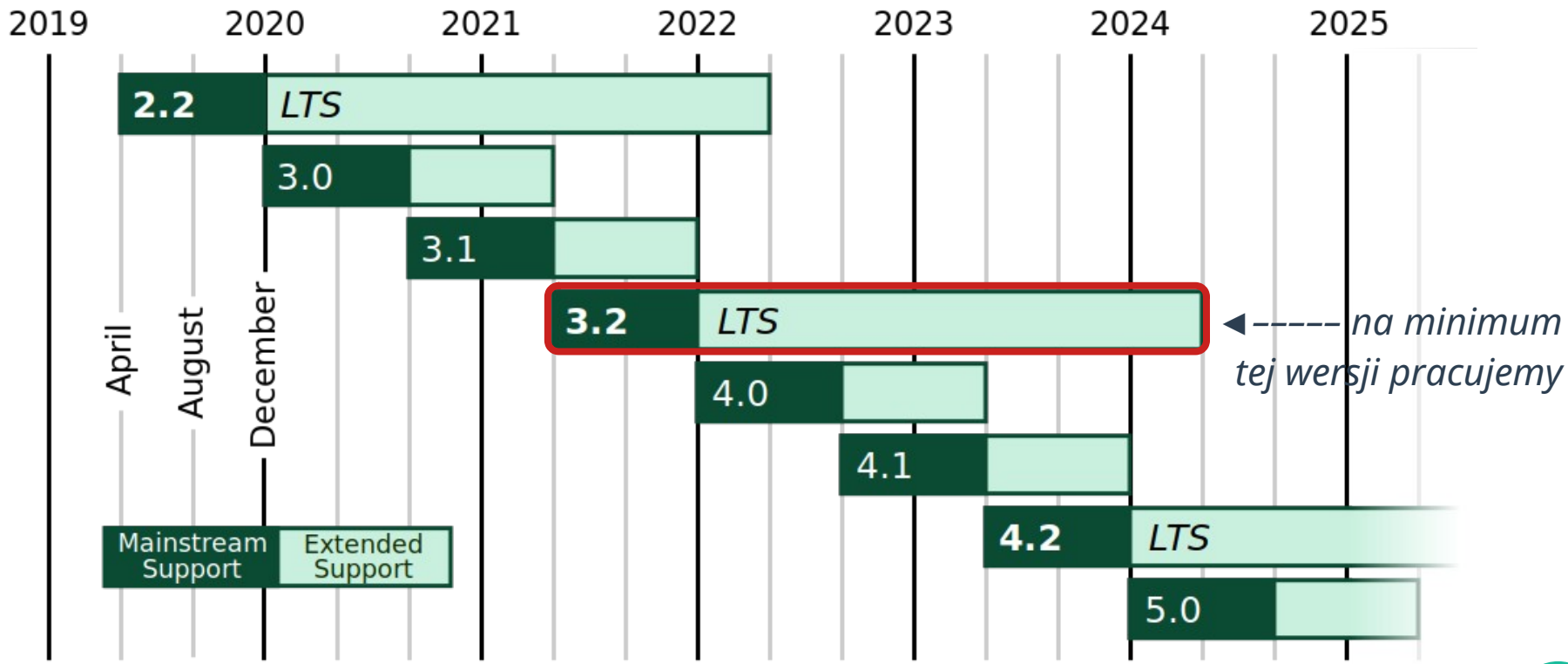
- współpraca z **serwerami HTTP** poprzez:
 - moduły ASGI/WSGI*: *Apache, Nginx*
 - moduł FastCGI: *Lighttpd*
- współpraca z backendem **bazodanowym**:
 - Oficjalnie: *PostgreSQL, MySQL, MariaDB, SQLite* oraz *Oracle*
 - Można użyć: *MS-SQL, IBM Db2, SQL Anywhere* oraz *Firebird*
 - Forki: *NoSQL, MongoDB, Google App Engine*

***ASGI** – Asynchronous Server Gateway Interface

***WSGI** – Web Server Gateway Interface

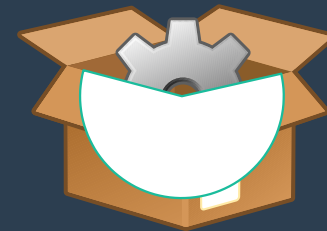
Wersje frameworka

Instalacja i Uruchomienie



Instalacja modułu Django

Instalacja i Uruchomienie



1) instalacja interpretera Python

2) instalacja modułu Django

- w katalogu użytkownika
- dowiązanie do *site-packages*
- pobranie i wypakowanie do wybranego katalogu oraz:
`python setup.py install`

3) założenie katalogu projektu

- **gdzie?**

Instalacja

```
# python -m pip install Django
```

Weryfikacja

```
$ python -m django --version
```

```
$ python
```

```
>>> import django
```

```
>>> print(django.get_version())
```


Umieszczenie projektu

Katalogi i PyPodejście



Język PHP oraz Witryny Web przyzwyczały nas do umieszczania Projektów w katalogu WWW Apache/Nginx:



`/var/www/<proj-name>/`

Web aplikacje Django oczekują umieszczenia w katalogu **INNYM** niż WWW Apache/Nginx, np. w **katalogu użytkownika**:

`/home/<USER>/<proj-name>/`



Wybierz katalog dla projektu ...

... zainstaluj moduł Django

3) Załóż projekt:

```
$ django-admin startproject <project-name>
```

Struktura projektu

Katalogi i PyPodejście



```
<proj-name>/      1
  manage.py        2
  <proj-name>/      3
    __init__.py     4
    settings.py     5
    urls.py          6
    asgi.py          7
    wsgi.py          8
```

- 1 **<proj-name>** – kontener projektu
- 2 **manage.py** – polecenie zarządzania projektem
- 3 **<proj-name>** – główny katalog projektu – nazwa pakietu Python dla poleceń importu
- 4 **__init__.py** – pusty plik informujący Pythona, że ten katalog ma być uważany jako pakiet
- 5 **settings.py** – ustawienia dla projektu Django
- 6 **urls.py** – deklaracje URL (router/TOC) dla projektu
- 7 **asgi.py** – punkt wejścia dla serwerów hostujących projekt, kompatybilnych z ASGI
- 8 **wsgi.py** – j/w dla WSGI

Start dev-serwera

Instalacja i Uruchomienie

Uruchomienie serwera:

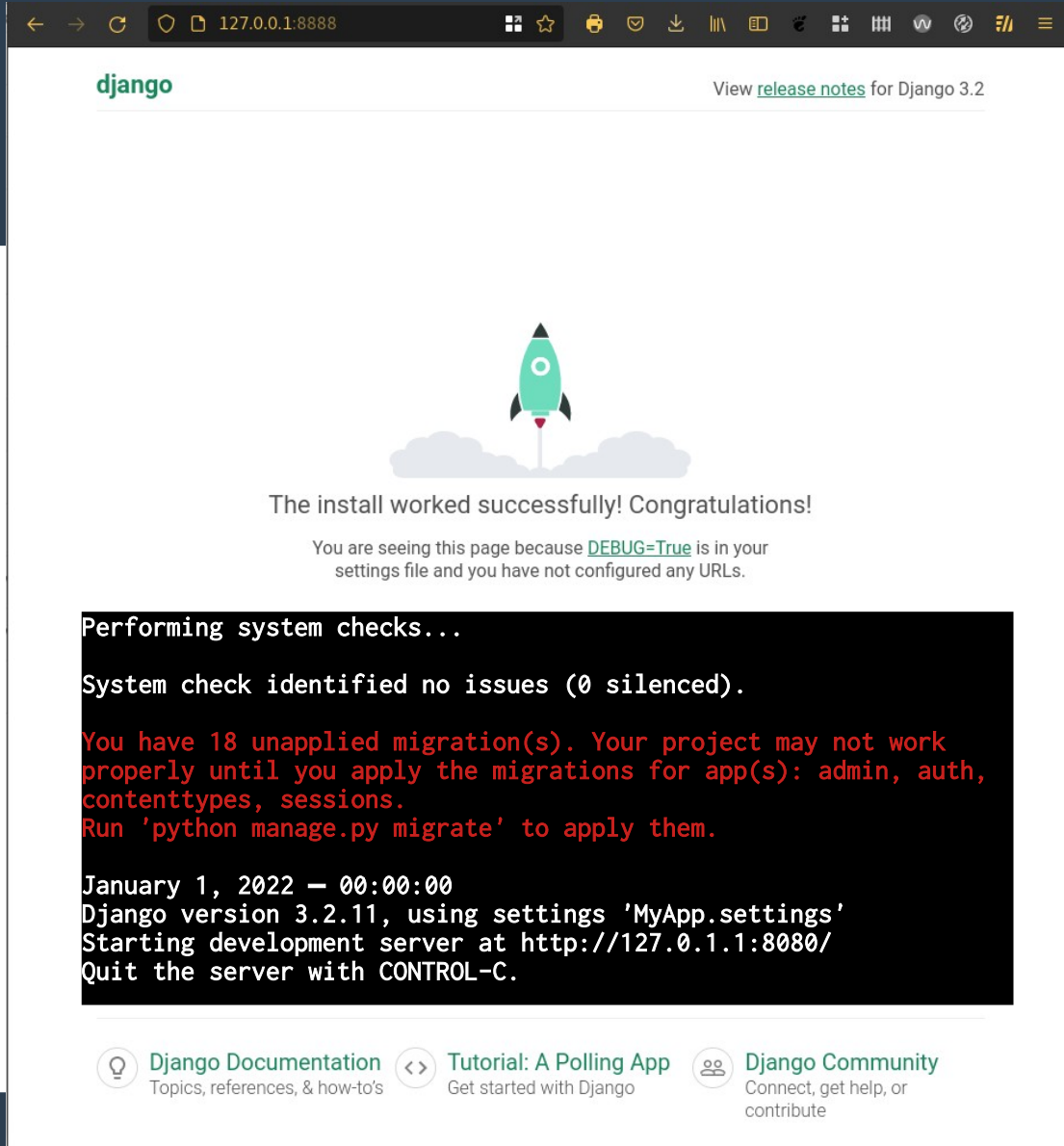
```
$ python manage.py runserver
```

Zmiana portu

```
(..) runserver 8080
```

Zmiana adresu serwera

```
(..) runserver 127.0.1.1:8080
```



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8888`. The page title is "django" and there is a link to "View [release notes](#) for Django 3.2". The main content area features a green rocket icon launching from a cloud. Below the icon, the text reads: "The install worked successfully! Congratulations!". A message follows: "You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs." Below this is a black terminal window with white text showing the following output:

```
Performing system checks...
System check identified no issues (0 silenced).
You have 18 unapplied migration(s). Your project may not work
properly until you apply the migrations for app(s): admin, auth,
contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

January 1, 2022 — 00:00:00
Django version 3.2.11, using settings 'MyApp.settings'
Starting development server at http://127.0.1.1:8080/
Quit the server with CONTROL-C.
```

At the bottom of the page, there are three links: "Django Documentation" (Topics, references, & how-to's), "Tutorial: A Polling App" (Get started with Django), and "Django Community" (Connect, get help, or contribute).

Uruchamianie dev-serwera

Projekty i Aplikacje

Projekt

- Kolekcja Ustawień i Aplikacji dla konkretnej witryny
- Projekt może zawierać wiele Aplikacji

Aplikacja

- Web-aplikacja wykonuje jakieś działania
- Aplikacja może być w wielu Projektach

```
def questions():  
    return answer
```