II. VYTVOŘENÍ NOVÉ APLIKACE, ZÁKLADNÍ KONFIGURACE, URL MAPOVÁNÍ

1. Vytvoření nové aplikace

Každý web vyvíjený ve frameworku Django je tvořen z tzv. **aplikací**. Součástí webu může být několik aplikací (např. katalog, eshop, blog, galerie, chat atd.), minimálně aspoň jedna, jejíž úvodní stránka bývá zároveň vstupní (domovskou) stránkou celého webu.

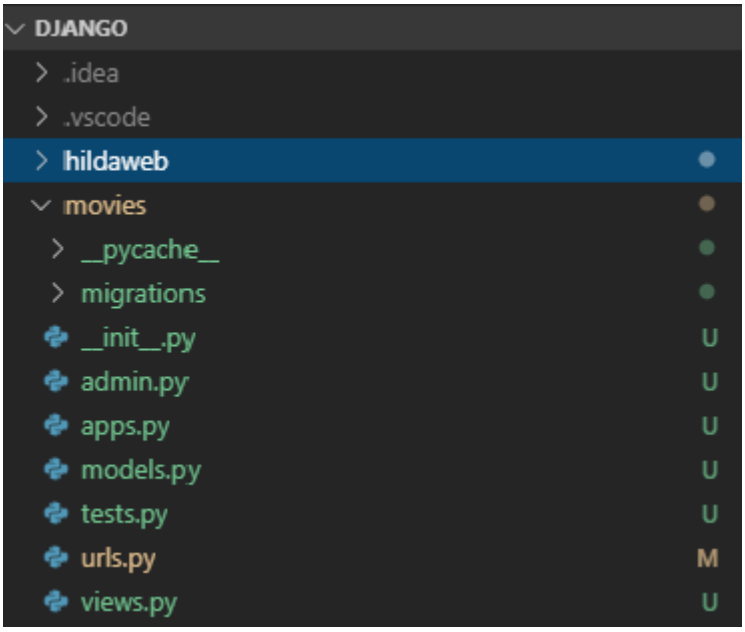
V našem případě vytvoříme "startovací" aplikaci zaměřenou na databázi filmů a nazveme ji movies. Použijeme manažer **manage.py** a příkaz **startapp**. Příkaz zadáváme ze základního adresáře našeho webu, tedy ze složky **django**:

```
py manage.py startapp movies
```

*Všimněte si, že interpret můžeme v novějších verzích jazyka spouštět místo příkazu python příkazem **py**. Od verze Python 3.7.0 je to ve Windows dokonce doporučená varianta.*

Podrobnější informace naleznete zde: https://docs.python.org/3/using/windows.html#launcher

Po zadání příkazu se ve struktuře webu objeví nová složka **movies** obsahující několik důležitých souborů:



2. Konfigurace webu (soubor hildaweb\settings.py)

Další důležité kroky provedeme v konfiguračním souboru našeho webu settings.py.

Podrobná dokumentace k souboru settings.py: https://docs.djangoproject.com/en/3.0/ref/settings/

Nejprve přidáme aplikaci movies do pythonovského seznamu (listu) k dalším již předinstalovaným aplikacím, které jsou standardní součástí frameworku Django. Řeší některé běžné a univerzální operace, jakými jsou administrace webu, autentizace uživatelů, typy obsahu, obsluha sessions (proměnné na straně serveru), informační zprávy nebo obsluha statických souborů (stylů, skriptů, příloh...). Do seznamu vepíšeme název naší aplikace - movies.

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'movies',
]
```

Musíme si dát pozor na správný zápis řetězce (uvozovky), na odsazení i čárku.

Dále v souboru settings.py zkontrolujeme nastavení databáze. V tomto okamžiku ještě nemusíme provádět žádné změny, protože prozatím budeme používat výchozí jednoduchou SQL databázi zvanou **SQLite**. Tato "odlehčená" relační databáze je použitelná pro jednoduché databázové aplikace, u nichž nepředpokládáme vysokou zátěž ani větší množství uživatelů. Výhodou je, že se nemusíme instalovat nějaký externí databázový systém a že všechna data budou uložena v souboru **db.sqlite3**.

```
# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Nakonec ještě nastavíme časovou zónu a jazyk pro náš web.

```
# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'cs'

TIME_ZONE = 'Europe/Prague'

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

Přehled identifikátorů pro jednotlivé jazyky světa: http://www.i18nguy.com/unicode/language-identifiers.html)

3. URL mapování (soubory urls.py)

V této podkapitole se budeme zabývat mapováním URL (URL mapping) v rámci našeho webu.
Adresování neboli **směrování** (**routing**) požadavků, které přicházejí z klientských prohlížečů na webový server, patří k tomu základnímu a velmi důležitému, co bychom měli během vývoje webové aplikace vyřešit. Webové frameworky tuto činnost programátorům obvykle výrazně usnadňují a zpřehledňují.
V Django se k URL mapování používají soubory s názvem **urls.py**. Kromě základního mapovacího souboru urls.py, který najdeme v základní složce webu (v tomto případě hildaweb), můžeme vytvářet samostatné stejnojmenné soubory v každé aplikaci, které odděleně mapují URL pro potřeby této aplikace.
Nejprve tedy provedeme nutné úpravy v hlavním souboru **hildaweb\urls.py**:

```
# Import modulu admin z balíku django.contrib
from django.contrib import admin
# Import modulu path z balíku django.urls
from django.urls import path
```

```
# Import metody include() - pomocí ní připojíme cesty nastavené v souboru urls.py, který je součástí aplikace movies
from django.urls import include
# Třída RedirectView, která je součástí balíku django.views.generic umožňuje přesměrování pohledů (stránek)
from django.views.generic import RedirectView
# Z balíku django.conf importujeme nastavení (settings)
from django.conf import settings
# Z balíku django.conf.urls.static importujeme metodu static(),
# pomocí které zajistíme obsluhu statických souborů (jen ve vývojové fázi aplikace)
from django.conf.urls.static import static

# Mapování URL adres webu - propojení URL s jejich obsluhou
urlpatterns = [
    # URL "admin/" je namapováno na administrační rozhraní webu
    path('admin/', admin.site.urls),
    # URL "movies/" je namapováno na aplikaci movies a její vlastní soubor 'movies.urls'
    path('movies/', include('movies.urls')),
    # domovská stránka webu ('') je automaticky přesměrována na url z předešlého řádku ('movies')
    path('', RedirectView.as_view(url='movies/')),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
# Metodou static() zajistíme správné mapování adres statických souborů webu (css, js, obrázků atd.).
# Jejich umístění bude definováno konstantami STATIC_URL a STATIC_ROOT v souboru settings.py
```

Pomocí komentářů v kódu si můžete udělat představu, jak mapování URL funguje. Jeho základem je seznam (list) URL "vzorků" uložený do proměnné **urlpatterns**. V něm se vždy určitá URL adresa spojuje s nějakou obsluhou, nejčastěji metodou, která má být na základě tohoto URL požadavku vyvolána. V některých případech může být obsluha požadavku "delegována" na jiné místo v aplikaci. Například v našem případě dojde po zadání "složky" **admin** za adresu webového serveru k předání řízení vestavěnému administračnímu rozhraní (**admin.site**), které má svůj vlastní modul **urls**.

Podobně to zařídíme i v případě naší aplikace **movies**, kterou jsme před chvílí začali vytvářet. Proto nyní ve složce **movies** založíme nový soubor nazvaný **urls.py** (**movies\urls.py**) a vložíme do něj tento obsah:

```
# Import modulu path z balíku django.urls
from django.urls import path
# Import modulu views (soubor views.py) ze stejné složky (.)
from . import views

# URL mapování - seznam URL adres pro aplikaci movies
urlpatterns = [

]
```

Jak sami vidíte, prozatím je jeho seznam **urlpatterns** prázdný. Za okamžik zjistíme, že to bude mít své důsledky...

4. Migrace a test stavu webu

Uděláme-li v aplikaci nějaké významnější změny, je vždy vhodné vytvořit tzv. **migrace** a poté je aplikovat na připojenou databázi. Zadáme postupně tyto dva příkazy:

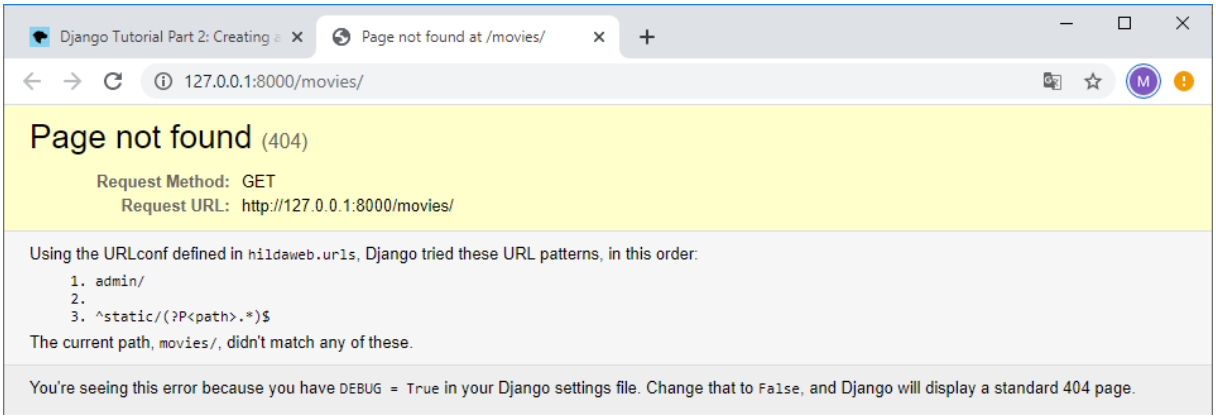
```
py manage.py makemigrations
```

```
py manage.py migrate
```

A nyní můžeme ověřit funkčnost a stav webového serveru:

```
py manage.py runserver
```

Klikneme-li na odkaz s lokální adresou, pod kterou nám webový server běží, ukáže se v této fázi tato chybová stránka:



Nemusíme se děsit, vše je vlastně v pořádku. Všimněme si, že došlo k automatickému přesměrování na adresu 127.0.0.1:8000/movies tak, jak jsme to nastavili v základním mapovacím souboru **hildaweb\urls.py** v řádku:
`path('', RedirectView.as_view(url='movies/'))`

Obsluhu poté převzal soubor **movies\urls.py**, který však ještě žádné mapování URL neobsahuje. Proto server nahlásí chybu 404. Mějte, prosím, trpělivost, napravíme to v jedné z dalších kapitol.