Projet 13

Version 1.0

BICHARI Mehdi

janv. 08, 2021

Overview

CHAPITRE 1

Installation

1.1 Python

In order to install OCR-Projet5 you need to use python 3.8+.

If you don't already have it, please refer to python's site.

1.2 Python's modules

The modules needed by python are listed in the requirements.txt at the root of the project.

To install module, place yourself at the root of Projet13 and use the command :

```
pip3 install -r requirements.txt
```

For further explanation, see the « DE - P13 » file

CHAPITRE 2

Accounts

Models for accounts

class accounts.models.**AwaitingData**(**args*, ***kwargs*)

class to store data awaiting validation

exception DoesNotExist

exception MultipleObjectsReturned

class accounts.models.**CustomUser**(**args*, ***kwargs*)

Make user's email unique

exception DoesNotExist

exception MultipleObjectsReturned

class accounts.models.**Friends**(**args*, ***kwargs*)

class to make link between friends

exception DoesNotExist

exception MultipleObjectsReturned

Urls for accounts

Views for accounts

class accounts.views.**FriendsView**(***kwargs*)

Load friends page

form_class

alias de *accounts.forms.SearchFriendForm*

form_valid(*form*)

Action after form's validation

get_context_data(***kwargs*)

Construct context for template

get_form_kwargs()

Add user to form's kwargs

get_queryset()

Get Friends in db

model
alias de *accounts.models.Friends*

accounts.views.**check_mail**(*request*) → django.http.response.HttpResponse
Check if mail is integration database.

Paramètres **request** – django's request

Renvoie HttpResponse

accounts.views.**delete_friend**(*request*, *friend : accounts.models.CustomUser*) → django.http.response.HttpResponse
Delete a friend from friend's list

Paramètres
- **request** – django's request
- **friend** – friend to delete

Type renvoyé HttpResponse

accounts.views.**get_user_info**(*request*) → django.http.response.HttpResponse
View to send login data.

Paramètres **request** – django request

Renvoie HttpResponse

accounts.views.**login_user**(*request*, *form : accounts.forms.LoginForm*) → django.http.response.HttpResponse
Verify login.

Paramètres
- **request** – django request
- **form** (LoginForm) – form to retrieve login data

Renvoie HttpResponse

accounts.views.**reset_password**(*request*, *user : str*) → django.http.response.HttpResponse
Reset user's password.

Paramètres
- **request** – django's request
- **user** (*str*) – user to change password

Renvoie HttpResponse

accounts.views.**send_reset**(*request*, *user : str*) → django.http.response.HttpResponse
Send email to user to change password.

Paramètres
- **request** – django's request
- **user** (*str*) – user to change password

Renvoie

accounts.views.**sign_out**(*request*) → django.http.response.HttpResponse
View to log out.

Paramètres **request** – django request

Renvoie HttpResponse

accounts.views.**subscribe**(*request*) → django.http.response.HttpResponse
View to subscribe for a new user.

Paramètres **request** – django request

Renvoie HttpResponse

accounts.views.**validate**(*request*, *guid : str*) → django.http.response.HttpResponse
Validate data through a link received by mail.

Paramètres
- — **`request`** – django's request
- — **`guid`** – Guid to identify request sent

Renvoie HttpResponse

Forms for accounts

class `accounts.forms.`**`ChangePasswordForm`**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Class to create form integration order to change password

`clean`()

Get cleaned data

class `accounts.forms.`**`CheckMailForm`**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Class to create form to check is email integration db

class `accounts.forms.`**`LoginForm`**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Class to create form for user.

class `accounts.forms.`**`SearchFriendForm`**(**args*, ***kwargs*)

Class to create form in order to look for a friend

class `accounts.forms.`**`SubscribeForm`**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Class to create form for creating user.

`clean`()

Get cleaned data

Commands for accounts

`accounts.commands.commands.`**`mail_password`**(*request*, *user : str*)

Send mail to reset password.

Paramètres **`user`** (*`str`*) – user to reset password

`accounts.commands.commands.`**`mail_subscription`**(*request*, *data : Dict*) → None

Send mail for subscription.

Paramètres
- — **`request`** – django's request
- — **`data`** (*`Dict`*) – data sent integration subscription

Type renvoyé None

`accounts.commands.commands.`**`validate_subscription`**(*request*, *guid : str*) → None

Create account after validation.

Paramètres
- — **`request`** – django's request

— **`guid`** – Subscription guid

Type renvoyé None

Purge data above 24h

class `accounts.management.commands.purge_awaiting_data.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Purge command

`handle`(**args*, ***options*)

Handle command

CHAPITRE 3

Library

Library's classes

class library.models.**Game**(**args*, ***kwargs*)
  Games

  exception DoesNotExist

  exception MultipleObjectsReturned

class library.models.**LendedGame**(**args*, ***kwargs*)
  lended games

  exception DoesNotExist

  exception MultipleObjectsReturned

class library.models.**OwnedGame**(**args*, ***kwargs*)
  Games owned by user

  exception DoesNotExist

  exception MultipleObjectsReturned

class library.models.**Platform**(**args*, ***kwargs*)
  Platforms for games

  exception DoesNotExist

  exception MultipleObjectsReturned

class library.models.**WantedGame**(**args*, ***kwargs*)
  Wanted games

  exception DoesNotExist

  exception MultipleObjectsReturned

library.models.**handle_return_date**(*sender*, ***kwargs*)
  Actualize return date if returned

Urls for accounts

Views for library

class library.views.**BorrowedView**(***kwargs*)
  Load borrowed games

`get_context_data`(***kwargs*)
Construct context

`get_queryset`()
Get borrowed games

`model`
alias de *`library.models.LendedGame`*

`class` `library.views.`**`GameListView`**(***kwargs*)
Load owned games

`form_class`
alias de *`library.forms.LendGameForm`*

`form_valid`(*form*)
Action if form is valid

`get_context_data`(***kwargs*)
Construct context

`get_form_kwargs`()
Get user in form

`get_queryset`()
Get owned games

`model`
alias de *`library.models.OwnedGame`*

`class` `library.views.`**`WantedView`**(***kwargs*)
Load wanted games

`get_context_data`(***kwargs*)
Construct context

`get_queryset`()
Get wanted games

`model`
alias de *`library.models.WantedGame`*

`library.views.`**`add_to_library`**(*request*, *game_ : str*) → django.http.response.HttpResponseRedirect
Add a game to user's library.

Paramètres
— **`request`** – django's request
— **`game`** – game to add to library

Type renvoyé HttpResponseRedirect

`library.views.`**`add_wish`**(*request*, *game_ : str*) → django.http.response.HttpResponseRedirect
Add a game to user's wish list.

Paramètres
— **`request`** – django's request
— **`game`** – game to add to wish list

Type renvoyé HttpResponseRedirect

`library.views.`**`delete_from_library`**(*request*, *owned_game : library.models.OwnedGame*) → django.http.response.HttpResponseRedirect
Delete a game from user's library.

Paramètres
— **`request`** – django's request
— **`owned_game`** – game to delete from the library

Type renvoyé HttpResponseRedirect

`library.views.`**`delete_wish`**`(`*request*, *wanted_game : library.models.WantedGame*`)` → django.http.response.HttpResponseRedirect

Delete a game which was in wish list.

Paramètres
- **`request`** – django's request
- **`wanted_game`** – game to delete from wish list

Type renvoyé HttpResponseRedirect

`library.views.`**`unmark_lended`**`(`*request*, *lended_game : library.models.LendedGame*`)` → django.http.response.HttpResponseRedirect

Unmark a game which has been lended.

Paramètres
- **`request`** – django's request
- **`lended_game`** – game lended

Type renvoyé HttpResponseRedirect

Forms for library

`class` `library.forms.`**`LendGameForm`**`(`**args*, ***kwargs*`)`

class to create lending game form

`clean``()`

Make sure either borrower or unknown_borrower are filled

`class` `library.forms.`**`SearchGameForm`**`(`*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*`)`

Class to create form for user.

Library's commands

`library.commands.commands.`**`find_games`**`(`*query : str*, *query_platform : str*`)` → List[library.models.Game]

Find games based on platform.

Paramètres
- **`query`** – name of the game
- **`query_platform`** – game's platfomr

Type renvoyé List[*Game*]

`library.commands.commands.`**`get_platform`**`(`*game : Dict*, *query_platform : str*`)` → Optional[library.models.Platform]

Get platform for a given game.

Paramètres
- **`game`** – Dict representation of a game
- **`query_platform`** – game's platform

Renvoie

`library.commands.commands.`**`get_release_date`**`(`*game : Dict*`)` → str

Get release date for game if exists :param game : game's representation :return : release_date

Pass navbar to context_processors

`library.context_processors.navbar_search.`**`navbar_search`**`(`*request*`)`

Pass toolbar to all views

Decorator to handle navbar_search

`library.context_processors.navbar_search_decorator.`**`navbar_search_decorator`**(*function*)

Decorator to add navbar_search form by decorator

Populate platform

`class` `library.management.commands.populate_platforms.`**`Command`**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Populate platforms command

`handle`(**args*, ***options*)

Handle command

Get lended games

`library.templatetags.get_lended.`**`get_lended`**(*owned_game_id*) → Union[library.models.LendedGame, bool]

Get lended games.

Paramètres **`owned_game_id`** – id of game owned

Type renvoyé Union[*LendedGame*, bool]

CHAPITRE 4

gamelenders

gamelenders URL Configuration

The *urlpatterns* list routes URLs to views. For more information please see : https://docs.djangoproject.com/en/3.1/topics/http/urls/

Examples : Function views

1. Add an import : from my_app import views
2. Add a URL to urlpatterns : path("", views.home, name="home")

Class-based views

1. Add an import : from other_app.views import Home
2. Add a URL to urlpatterns : path("", Home.as_view(), name="home")

Including another URLconf

1. Import the include() function : from django.urls import include, path
2. Add a URL to urlpatterns : path("blog/", include("blog.urls"))

Accounts views

ASGI config for gamelenders project.

It exposes the ASGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/

WSGI config for gamelenders project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/

CHAPITRE 5

Scrapping

Construct request for giantbomb API

`scrapping.construct_requests.`**`add_data_format`**(*data_format : str*) → str

Add format to request.

Paramètres **`data_format`** (`str`) – format of data returned by the API

Type renvoyé str

`scrapping.construct_requests.`**`add_query`**(*query : str*) → str

Add query to request.

Paramètres **`query`** (`str`) – query to look for

Type renvoyé str

`scrapping.construct_requests.`**`add_resources`**(*resources : Union[List[str], str]*) → str

Add types of resources to request.

Paramètres **`List[str] resources`** (`str,`) – type of resources searched

Type renvoyé str

`scrapping.construct_requests.`**`construct_platform_request`**() → str

Construct platform request to populate the DB

`scrapping.construct_requests.`**`construct_request`**(**, query : str, data_format : str = 'json', resources : Union[List[str], str] = None*) → str

Construct request to search through giantbomb api.

Paramètres

— **`query`** (`str`) – query to look for
— **`data_format`** (`str`) – format of data returned by the API
— **`List[str] resources`** (`str,`) – type of resources searched

Type renvoyé str

Send request to giantbomb API

`scrapping.send_requests.`**`get_platforms`**()

Get platforms from GiantBomb

`scrapping.send_requests.`**`send_request`**(**, query : str, data_format : str = 'json', resources : Union[List[str], str] = None*) → List

Send request to giantbomb API.

Paramètres

— **`query`** (*`str`*) – query to look for
— **`data_format`** (*`str`*) – format of data returned by the API
— **`resources`** (*`List[str]`*) – type of resources searched

Type renvoyé List

CHAPITRE 6

Scripts

Scripts needed for installation.

Usages described in « DE - P13 ».

CHAPITRE 7

Indices and tables

— genindex
— modindex
— search

Index des modules Python

Index