Проект. Общие сведения. Назначение.

Проект представляет собой консольное приложение **sending.py**, состоящее из модулей, написанных на python.

Проект производит запросы на внешний REST API и загружает оттуда данные, после чего производит их выгрузку на целевой сервер пользователя.

Требования. Инструкции по использованию.

Для успешной работы этого проекта, на сервере пользователя необходимо установить django-app for_serializer, где в модуле serializers.py реализован универсальный класс сериализатора данных <path_to_project>.for_serializer.UnSerializer, позволяющий представлять данные любых моделей (path_to_project – путь к папке с проектом – прим.).

Для примера подключения django-app for_serialize, был реализован Проект сервера пользователя, находящийся в публичном репозитории:

https://github.com/skyshy0707/budgets

Для того, чтобы получить данные с сервера внешнего REST API, а затем выгрузить их на пользовательский, Вам потребуется запустить консольное приложение **sending.py** и передать ему необходимые параметры.

Чтобы узнать подробнее, о параметрах для sending.py, воспользуйтесь вызовом справки, введя в консоль:

>>**python sending.py** -h

Параметры можно задать и в файле config.ini и запустить sending.py без передачи параметров в консоли.

Задание параметров файле конфигурации config.ini (см. также комментарии в самом файле):

record_params. параметры для записи данных в БД пользователя:

model=budget.Budget

модель, относящаяся к целевому серверу пользователя, которая представляет таблицу данных, загружаемых с внеш. api

serialize_fields=status, code, name, startdate, enddate, parentcode, budgettype

поля в модели, заданной в **model** для сериализации:

renaming_fields={"budgtypecode": "budgettype"}

поля для переименовывания их имён в данных, полученных от внеш. api:

ключи --- в данных, значения --- в модели model, которые сервер пользователя переименует для передачи их на след. этап обработки

refilling_fields={"parentcode": {"map": "budget.Budget.code", "fill": "id"}}

поля в модели model, которые требуется заполнять значением отличным от значения в данных, полученных от внеш. сервера, найденных по тому же ключу

где ключ --- указывает на имя поля в модели **model**,
значение --- порядок заполнения такого поля:

> **'map'** --- указатель на модель, реализованную на целевом сервере пользователя и поле, по которому следует выбрать её экземпляр, значение поля находится в данных по ключу (см. выше описание ключа **refilling_fields**)
> **'fill'** --- поле в выбранном экземпляре модели, найденной по **'map'**, значение которого требуется извлечь для перезаполнения поля в модели **model**

record_method=create

метод записи данных, поступающих от внеш. сервера в пользовательскую бд:

create --- созд. новые строки данных, **update** --- модификация данных

external_api_params. параметры внешнего api (источник получения данных):

base_url=http://budget.gov.ru/epbs/registry/7710568760-BUDGETS/data?

url-адрес внеш. api

data_key=data

имя ключа, по которому находятся данные в ответе на get-запрос

headers={}

заголовки (при необходимости)

params={"filterstatus": "ACTIVE"}

параметры для get-запроса на получения данных

cli_api_params. параметры пользовательского api:

base_url=http://127.0.0.1:8090/budget

url-адрес целевого сервера пользователя:

headers={}

заголовки (при необходимости):

edit_instance_by_fields=code, name, budgettype

параметры, по которым выбирается объект для обновления данных в БД пользователя (это имена полей модели, заданной в model)

Пример выполнения команды к консоли cmd:

```
<<python sending.py --model "budget.Budget" --serialize_fields "status, code, name, startdate, enddate, parentcode, budgettype" --renaming_fields "{'budgtypecode': 'budgettype'}" --refilling_fields "{'parentcode': {'map': 'budget.Budget.code', 'fill': 'id'}}" --url "http://budget.gov.ru/epbs/registry/7710568760-BUDGETS/data?" --data_key data --headers "{}" --params "{'filterstatus': 'ACTIVE'}" --cli_url "http://127.0.0.1:8090/budget" --cli_headers "{}" --edit_instance_by_fields "code, name, budgettype" --record_method create
```

Дополнительно

При работе модуля **sending.py** в корне папки с проектом ведётся журнал ошибок bad_requests.log, если отправка данных на целевой сервер пользователя вызвала ошибки с кодом в интервале целых чисел [400, 599].