Python Practice

Добавление курса BTC. Динамический импорт модулей

Автор курса

Кременtарь Ксения

Ведущий Python разработчик

Системный архитектор

в компании K-Solutions

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Добавление курса BTC. Динамический импорт модулей

На этом уроке

1. Добавим новые курсы в базу данных курсов , а именно: BTC => UAH, BTC => RUB, BTC => USD.
2. Внесем изменения в скрипт privat_api, для возможности сохранять курс не только USD, но и BTC относительно доллара.
3. Добавим получение курсов BTC => RUB и BTC => UAH с помощью сервиса https://www.cryptonator.com/api/
4. Добавим динамический импорт модулей api для получения различных курсов.
5. Проверим корректность работы проекта после внесенных изменений.

Модификация модуля privat_api

Суть: добавить возможность работать с валютой BTC, а именно - получать курс BTC=> USD.

Переименуем метод find_usd_rate => find_rate и добавим передачу параметра, алиас валюты, курс которой нас интересует, в функцию find_rate для поиска любой валюты, а не только USD.

Для соответствия кода валюты алиасу из ответа по api удобно использовать словарь, например - privat_aliases_map. В качестве ключей — код валюты, из параметра from_currency. А значение — алиас валюты, поиск которой нужно осуществлять в ответе, полученном по api.

```
privat_aliases_map = {
    840: "USD",
    1000: "BTC"
}
```

Защита от KeyError

Наверное, наиболее частая ошибка при работе со словарями — это KeyError.
Из официальной документации — исключение KeyError генерируется, когда ключ не найден в словаре, при использовании оператора доступа по ключу [].

```
>>> logins_map = {"Kseniia": "ksu18", "Alexey": "alex74"}
>>> name = "Vitaliy"
>>> login = logins_map[name]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Vitaliy'
```

Как избежать необработанного исключения?

1. Использовать метод get и значение по умолчанию, если логика позволяет.
2. Явно проверять в коде, есть ли ключ в словаре и реализовывать различную логику поведения, если значение по умолчание не может быть использовано.

В нашем случае, будем явно проверять, что словарь privat_aliases_map содержит ключ с интересующей нас валютой. Если нет — то генерировать явную ошибку, управляемую нами.

Использование метода get

```
logins_map = {"Kseniia": "ksu18", "Alexey": "alex74"}

name = "Kseniia"
login = logins_map[name]
print(login)  # выведет ksu18

name = "Vitaliy"

login = logins_map[name]  # выполнение прервется ошибкой KeyError: 'Vitaliy'

login = logins_map.get(name, "Undefined")
print(login)  # выведет Undefined

login = logins_map.get(name, name)
print(login)  # выведет Vitaliy
```

Реализация модуля cryptonator

Описание api: https://www.cryptonator.com/api

URL для отправки запросов:

- Для получения курса BTC=>UAH: https://api.cryptonator.com/api/ticker/btc-uah
- Для получения курса BTC=>RUB: https://api.cryptonator.com/api/ticker/btc-rub

HTTP метод: GET

Особенность api — различные URL для разных валют. Это необходимо учесть в скрипте cryptonator.

API cryptonator

Пример ответа:


```
{
    "ticker": {
        "base": "BTC",
        "target": "RUR",
        "price": "495563.14963645",
        "volume": "710.80011038",
        "change": "555.79921060"
    },
    "timestamp": 1536047943,
    "success": true,
    "error": ""
}
```

```
{
    "ticker": {
        "base": "BTC",
        "target": "UAH",
        "price": "183404.19577280",
        "volume": "129.86963291",
        "change": "328.26538567"
    },
    "timestamp": 1541517902,
    "success": true,
    "error": ""
}
```

Неудобство текущей реализации

При вызове обновления курса, нужно точно знать какой класс api создавать, то есть в каком модуле реализовано обновление для данного курса.

Но что, если мы перепутаем модуль? Давайте проверим. Но как? Напишем тест, конечно же!

Вызовем обновление курса BTC=>UAH с помощью модуля cbr_api, а USD=>RUB с помощью модуля privat_api.

Решение

Вывод — лучше переложить на систему golden-eye обязанность выбирать модуль, который нужно вызывать.

Каким образом это может быть сделано? Самый очевидный и простой вариант — хранить в базе данных, в таблице курсов, информацию о модуле, в котором реализовано обновление данного курса.

При вызове обновления курса, указывать только какой курс обновить — то есть from_currency и to_currency. А дальше система будет сама решать, какой именно модуль нужен.

Встроенный модуль importlib

Стандартная библиотека importlib позволяет импортировать модули динамически, то есть без явного указания инструкции import в коде. Функция importlib.import_module ожидает обязательным параметром название модуля и осуществляет импорт этого модуля, а в качестве результата выполнения - возвращает сам объект модуля. У этого объекта, как и при обычном импорте, доступны описанные в модуле классы, функции, переменные и т.п.
Второй, необязательный параметр функции importlib.import_module — это package, в котором находится модуль.

```
# в глобальной области видимости становится доступным имя cbr_api
import cbr_api

import importlib

# создается переменная cbr_api_dynamic, ее тип – модуль
cbr_api_dynamic = importlib.import_module("cbr_api")


print(cbr_api == cbr_api_dynamic)  # выведет True
```

Двигаемся дальше!

Итак, все готово для внесения изменений в код. Что именно нам нужно сделать?

- Добавить новое поле в таблицу курсов, module.
- Реализовать динамический импорт в коде проекта.
- Модифицировать тесты.

Что получилось к настоящему моменту

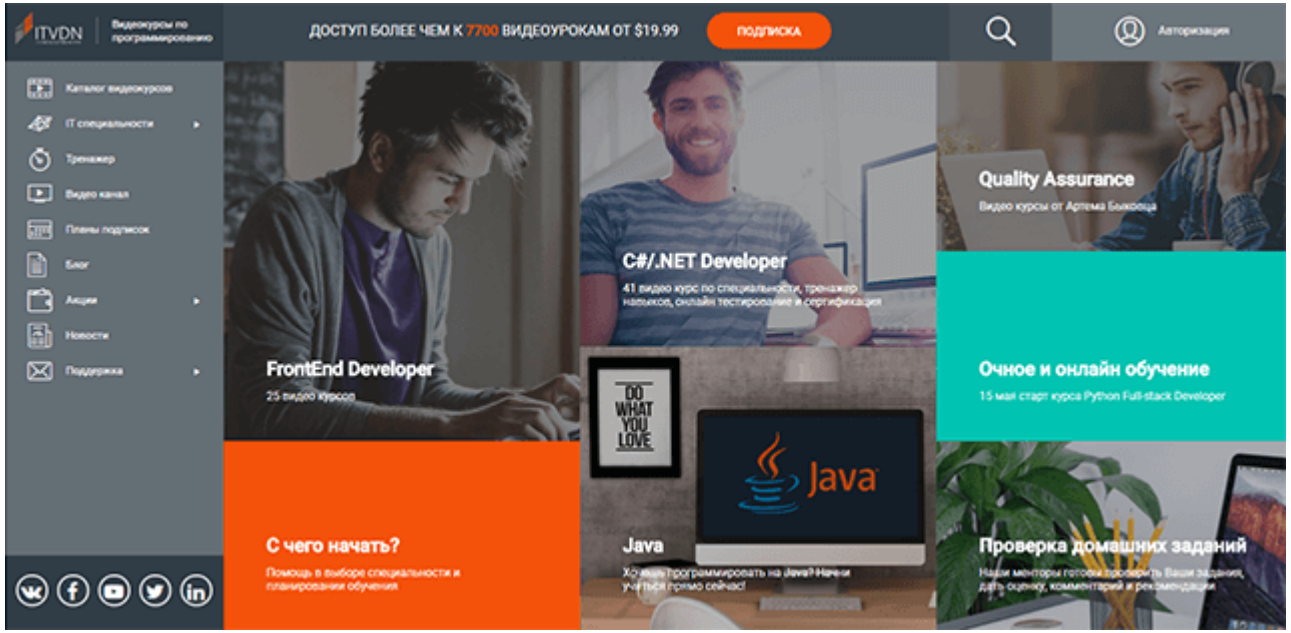
- Добавлен функционал по обновлению курса валют относительно валюты BTC.
- Ядро сервиса golden-eye готово!

Что дальше?

- Построение web сервиса на основе фреймворка.
- Добавление первой странички Hello world!
- Перенос ядра проекта в web сервис.

Смотрите наши уроки в видео формате

ITVDN.com



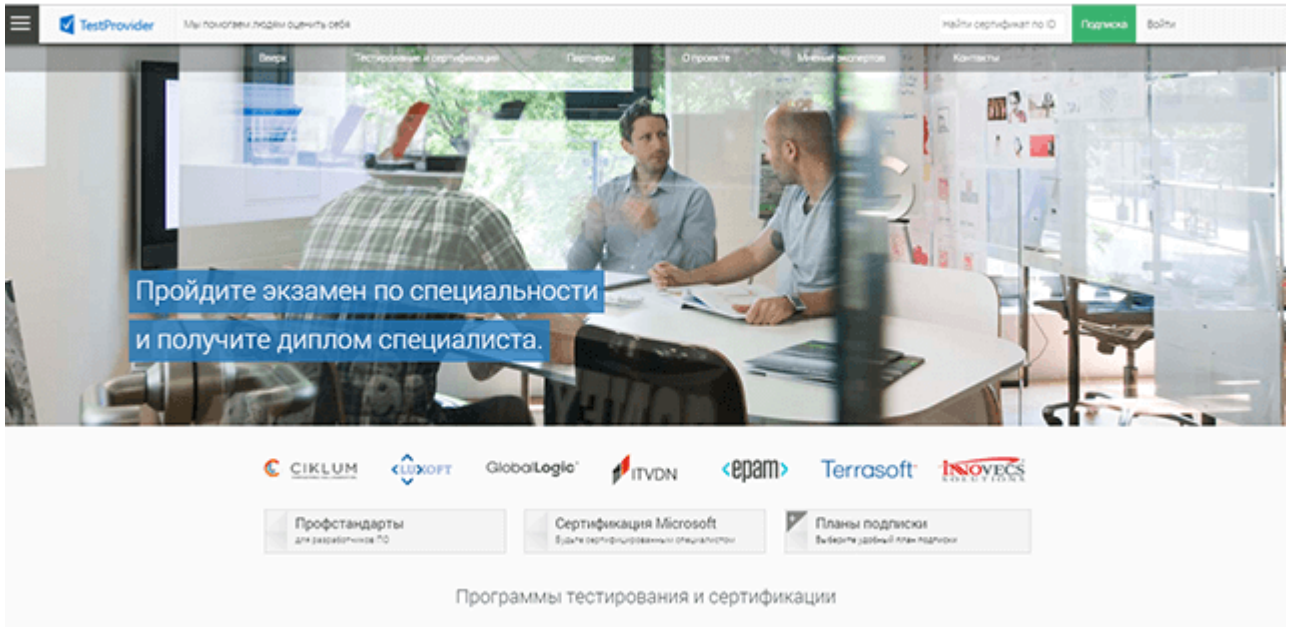
Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics и другими высококвалифицированными разработчиками.

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association