



# Python Practice

Добавление возможности обновления курсов с сайта и отображения логов

# Python Practice

## Автор курса



Крементарь Ксения

Ведущий Python разработчик

Системный архитектор

в компании K-Solutions

# Python Practice

После урока обязательно



Повторите этот урок в видео формате на  
[ITVDN.com](http://itvdn.com)



Проверьте как Вы усвоили данный материал на  
[TestProvider.com](http://testprovider.com)

Добавление возможности обновления курсов с сайта и отображения логов

# Python Practice

## На этом уроке

1. Добавим url, при вызове которого с сайта будет происходить обновление курсов с помощью пакета `api`.
2. Добавим новый контроллер и реализуем его логику.
3. Добавим на страницу отображения курсов url для обновления каждого курса по отдельности, а также для всех курсов.
4. Добавим отображение логов из БД с пагинацией.
5. Проверим внесенные изменения — вручную и с помощью тестов.

# Python Practice

## Добавление новой view-функции

При добавлении нового функционала в наше web приложение удобно начинать с описания view-функции и указания url, при котором она будет вызываться.

В нашем случае нас сейчас нужно добавить возможность обновлять либо все курсы из таблицы курсов, либо какой-то конкретный курс (с указанием `from_currency` и `to_currency`)

Сделать это можно конечно же многими способами, но опять же, чтоб не писать много похожего кода, можем сделать логику следующей — если в url указаны `from_currency` и `to_currency`, то будет происходить обновление конкретного курса. Если не указаны ни `from_currency` ни `to_currency`, обновление всех курсов из БД.

# Python Practice

## Выберем принцип построения url

Тогда url может иметь вид —

`/update/<int:from_currency>/<int:to_currency>` или `/update/all`,

и оба они будут обрабатываться одной и той же view функцией

```
@app.route('/update/<int:from_currency>/<int:to_currency>')
```

```
@app.route('/update/all')
```

```
def update_xrates(from_currency=None, to_currency=None):
```

```
    return f"update by api"
```

А уже обработке запроса проверять, переданы ли параметры `from_currency`, `to_currency` и в зависимости от этого делать обновление для каждого курса или только для выбранного.

# Python Practice

## Логика нового контроллера

Логика работы нового контроллера простая — в зависимости от переданных параметров либо делать запрос в БД и по списку вызывать `api.update_rate` с каждым курсом, или просто вызывать `api.update_rate`, передав параметры `from_currency`, `to_currency` из `url`.

Все остальное за нас сделает пакет `api`, реализованный в первой части курса.

Но что же делать после успешного обновления по `api`? Логично было бы отображать страницу с курсами — можно вызвать `render_template`, но для этого нам нужно опять получить данные о курсах из БД и т.п.

К тому же в приложении уже есть `url`, в котором реализовано отображение курсов — лучше используем Flask функцию `redirect`, которая позволяет перенаправлять пользователя на указанный в аргументе `url`.



# Python Practice

## Проверим в браузере... Работает!

После того, как готов url и реализована ее функция-обработчик, мы даже можем уже его проверить в браузере, введя в адресной строке:

`http://localhost:5000/update/840/980` или  
`http://localhost:5000/update/all`

Но ведь это не очень удобно, каждый раз вбивать url вручную — да и цель у нас — добавить отображение кнопки или url на странице отображения курсов в нашем приложении.

Для этого нужно внести изменения в шаблон, указать в таблице url для обновления каждого курса. И сделать это удобно с помощью функции Flask `url_for`.

# Python Practice

## Функция url\_for

Во Flask есть возможность динамического построения url с помощью `url_for` функции.

В качестве обязательного аргумента при вызове `url_for` необходимо передать название view-функции, для которой и нужно сгенерировать url. Также могут быть переданы именованные аргументы, которые учувствуют в формировании url.

Например, для функции `update_xrates`

```
url_for('update_xrates') => /update/all
```

или

```
url_for('update_xrates', from_currency=840, to_currency=980) => /update/840/980
```

или

```
url_for('update_xrates', new_arg=12) => /update/all?new_arg=12
```

# Python Practice

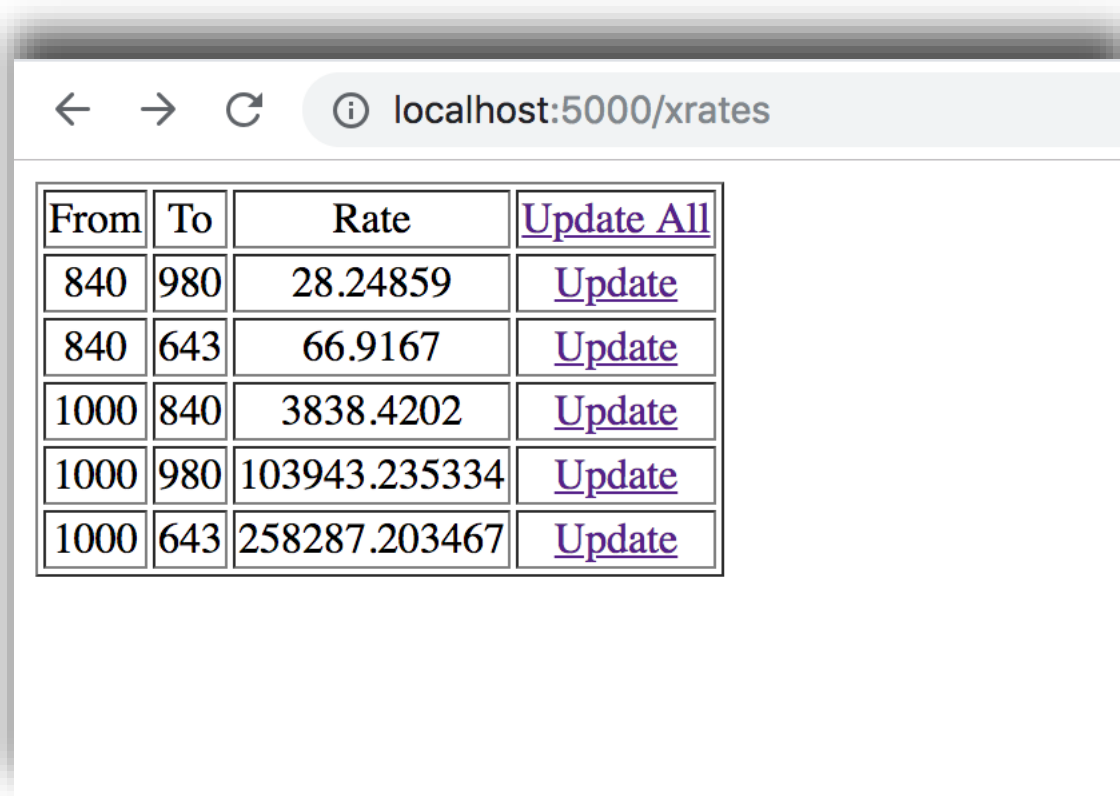
## Изменения в шаблоне xrates.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Rates</title>
</head>
<body>
  {% if not xrates %}
  <h1>No rates!</h1>
  {% endif %}

  <table border="1">
    <thead align="center">
      <td>From</td><td>To</td><td>Rate</td><td><a href="{{url_for('update_xrates')}}">Update All</a></td>
    </thead>
    {% for rate in xrates %}
    <tr align="center">
      <td>{{rate.from_currency}}</td>
      <td>{{rate.to_currency}}</td>
      <td>{{rate.rate}}</td>
      <td>
        <a href="{{url_for('update_xrates', from_currency=rate.from_currency, to_currency=rate.to_currency)}}">Update</a>
      </td>
    </tr>
    {% endfor %}
  </table>
</body>
</html>
```

# Python Practice

## Вид страницы курсов в браузере



A screenshot of a web browser window. The address bar shows 'localhost:5000/xrates'. The page content is a table with four columns: 'From', 'To', 'Rate', and a link column. The first row of the table has headers 'From', 'To', 'Rate', and a link 'Update All'. The following five rows contain data: (840, 980, 28.24859, Update), (840, 643, 66.9167, Update), (1000, 840, 3838.4202, Update), (1000, 980, 103943.235334, Update), and (1000, 643, 258287.203467, Update). The 'Update' links are purple and underlined.

From	To	Rate	<a href="#">Update All</a>
840	980	28.24859	<a href="#">Update</a>
840	643	66.9167	<a href="#">Update</a>
1000	840	3838.4202	<a href="#">Update</a>
1000	980	103943.235334	<a href="#">Update</a>
1000	643	258287.203467	<a href="#">Update</a>

## Этапы добавления отображения логов

1. Как обычно, выберем url для нового функционала и название view-функции.
2. Создать новый контроллер и реализовать его логику — получение логов из базы данных и рендеринг html шаблона.
3. Создание нового шаблона для отображения логов.

Но!

В ходе работы приложения количество логов будет расти, не очень удобно отображать все их на одной странице сайта.

Для решение этой проблемы, одно из лучших подходов — использование пагинации.

# Python Practice

## Пагинация

Пагинация — это способ отображения большого количества информации не целиком, а частями (страницами) с указанием номера страницы.

В Python этот подход может быть реализован разными способами, но в `peewee` есть отличный метод `paginate`. При вызове его у какого-либо объекта `select` запроса необходимо указать первым параметром номер страницы, вторым — количество элементов в выборке. Номер страницы может передаваться в аргументе `url` запроса — `page`.

Добавим использование метода `paginate` в контроллере по работе с логами, а также обработку параметра `page`.

## Тестирование внесенных изменений

Несмотря на то, что, что мы вносили изменения на html страничку, мы можем протестировать их не только вручную, но и с помощью тестов и библиотеки requests.

Для этого нам достаточно сделать вызов нужного нам url и распарсить полученный результат.

И проверить наличие нужных xml элементов. Что мы и сделаем — и завершим на сегодня.

# Python Practice

## Что дальше?

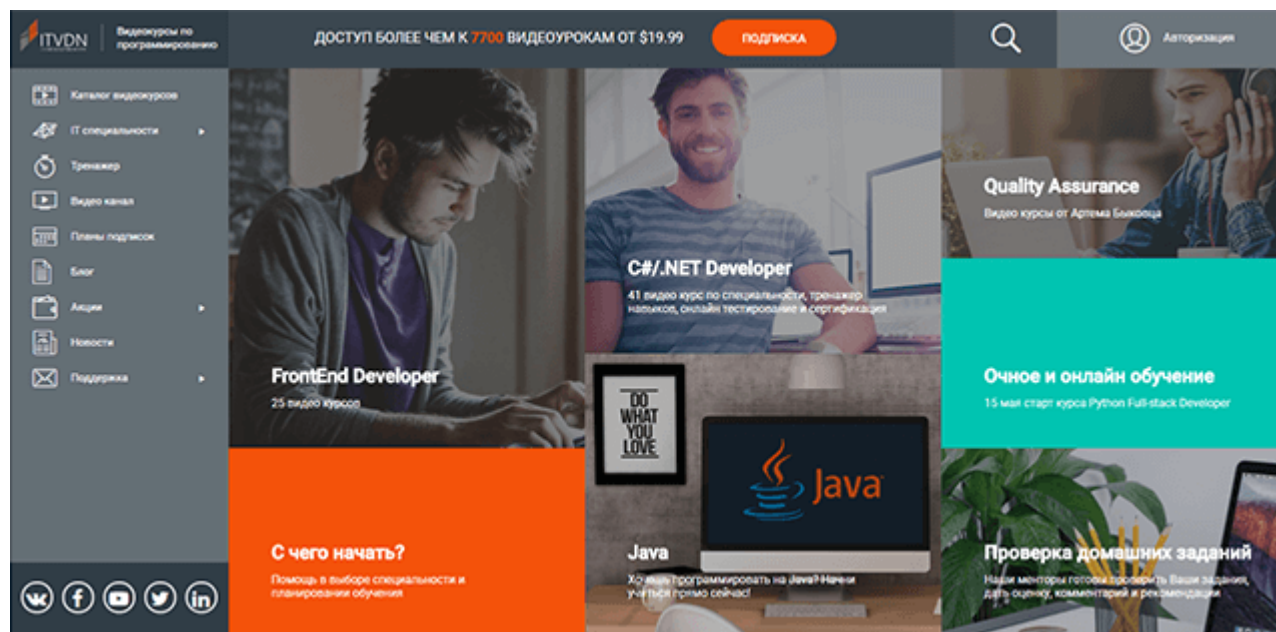
Добавление редактирования курсов вручную — знакомство с POST запросами и их обработкой во Flask.

Для того, чтоб защитить приложение Golden Eye и не допустить ручное обновление курсов всеми желающими, добавим простейшую защиту, проверку по ip — и сделаем это с помощью механизма декораторов в Python.



# Смотрите наши уроки в видео формате

ITVDN.com



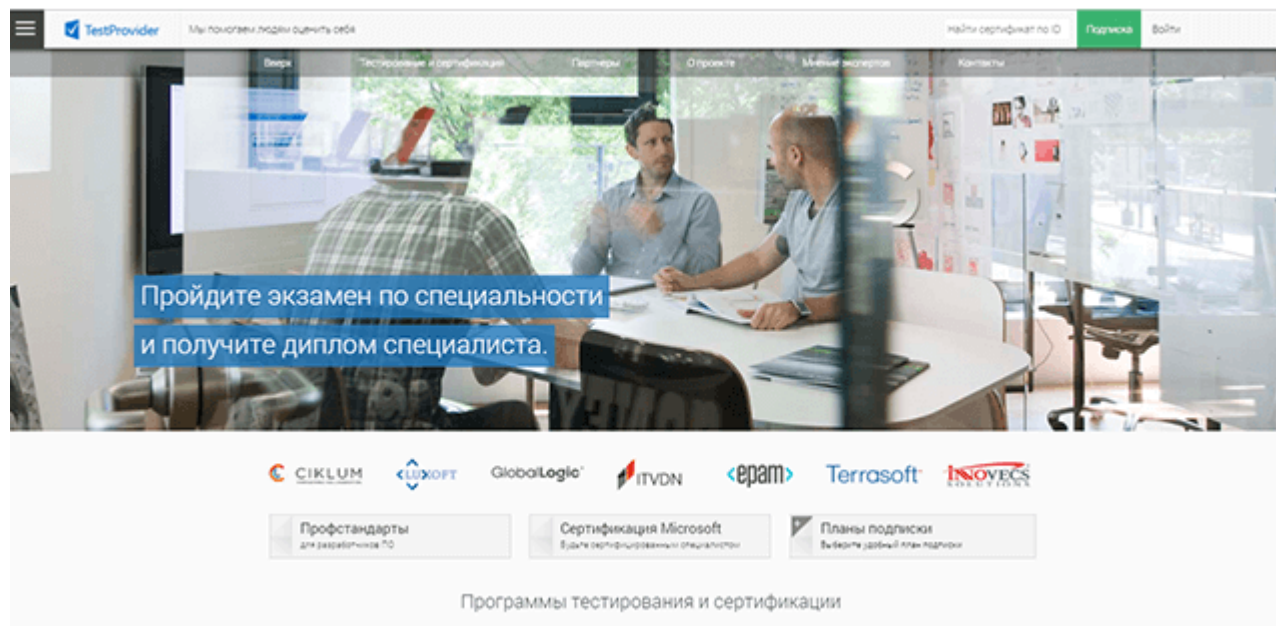
Посмотрите этот урок в видео формате на образовательном портале [ITVDN.com](http://ITVDN.com) для закрепления пройденного материала.

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



# 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](https://testprovider.com)

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.



# Python Practice

Q&A

# Информационный видеосервис для разработчиков программного обеспечения

