Сайт django

Задание 2

https://www.djangoproject.com/ -- это для напоминания, что остановились на этом

Сайт Блог

Следуйте данным инструкциям и запишите все ваши действия приблизительно как тут приведено ниже

1.Начальный стандартный WEB сервер Django

Перейдити в директорию **E:\NewEx** или создайте свою директорию (корневой диск E у вас может быть другим) и перейдите в неё, в которой разместите код блога. Для определенности изложения в тексте эта директория расположена на диске E и имеет имя NewEx. В данной директории установлена виртуальная среда **env_ex.** Вы можете виртуальную среду назвать как вам удобно. Активируйте виртуальную среду командой из командной строки (жирный шрифт использован для обозначения вводимых с клавиатуры команд)

E:\NewEx>**env_ex\scripts\activate**

Обычно команда указывается без перечисления пути к папке (как выше). В этом случае может быть написано так

$ **env_ex\scripts\activate (**или просто **env_ex\scripts\activate)**

Для удобства всегда будет указан полный путь.

Теперь можете начать Django проект, например, с именем **myfirst.** Здесь **myfirst** – это присвоенное пользователем имя проекта, которое может быть произвольным. Начало проекта задается исполнением такой команды

(env_ex) E:\NewEx>**django-admin startproject myfirst**

В результате в папке **NewEx** (в которой находится папка **env_ex)** будет создана папка **myfirst** со следующей структурой.

E:\NewEx\myfirst

```
|  manage.py
└──myfirst
    settings.py
    urls.py
    wsgi.py
    __init__.py
```

Это стандартный набор папок и файлов, который среда разработки web-приложений Django генерирует для помощи программисту. Как видно внутри внешней папки **myfirst** есть ещё одна внутренняя папка с тем же именем **myfirst**, что технически не удобно для указания пути к файлам при объяснении. Название внешней папки **myfirst** не играет никакой роли и может быть переименовано пользователем по его усмотрению.

Любым редактором переименуйте внешнюю папку **myfirst** в папку с именем, например, **blog** (это не обязательный технический прием, который устраняет повтор имен папок). Обычно внешнюю папку **myfirst** переименовывают в **myfirst_root**, подчеркивая значение корневой директории.

Перейдите в папку **blog** командой

(env_ex) E:\NewEx>**cd blog**

В командной строке запустите встроенный сервер разработки проекта следующей командой

(env_ex) E:\NewEx\blog>**python manage.py runserver**

Команда может быть набрана с сокращением **python** на **py**

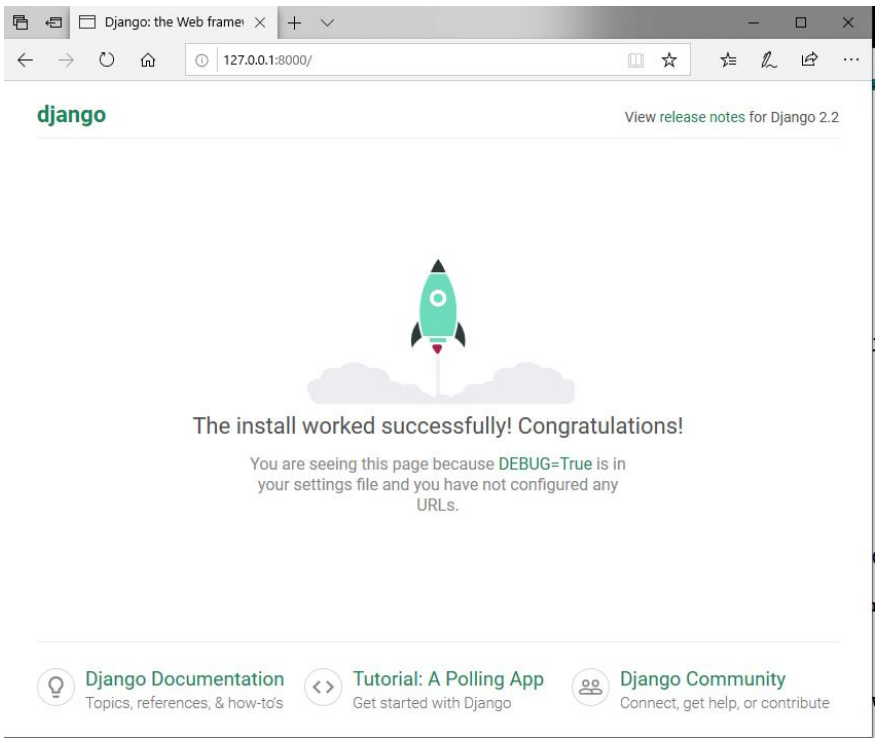
(env_ex) E:\NewEx\blog>**py manage.py runserver**

В окне командной строки отобразится сообщение похожее на

```
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
You have 17 unapplied migration(s). Your project may not
work properly until you apply the migrations for app(s):
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 21, 2019 - 13:19:26
Django version 2.2.6, using settings 'myfirst.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Сервер разработки Django – это встроенный упрощенный веб-сервер, который можно использовать при разработки сайта. Сервер разработки включен в состав Django чтобы создавать сайт, не отвлекаясь на настройку настоящего сервера до завершения работы. Сервер разработки автоматически перезагружается при внесении изменений в код файлов проекта о чем вы заметите позднее.

Если теперь в браузере, который установлен на вашем компьютере указать адрес http://127.0.0.1:8000/ то браузер отобразит начальный стандартный WEB сервер Django, который сгенерирован автоматически.



Данный сайт имеет реальные ссылки на сайт Django, документацию по Django и на сообщество пользователей. Все работает. В чем можно убедиться.

Для справки: в структуре файлов папки **Blog** автоматически произошло незначительное изменение, смысл которого будет пояснен позже. А именно папка **Blog** имеет теперь структуру состоящую из вложенных папок с именами (**myfirst, __pycahe__),** нескольких текстовых файлов Питон (расширение ***.py**), бинарных файлов с расширением ***.pyc** и файла базы данных **db.sqlite3**

E:\NewEx\Blog

```
        |  db.sqlite3
        |  manage.py
        └─myfirst
                |  settings.py
                |  urls.py
                |  wsgi.py
                |  __init__.py
                └────__pycache__
                        |  settings.cpython-37.pyc
                        |  urls.cpython-37.pyc
                        |  wsgi.cpython-37.pyc
                        |  __init__.cpython-37.pyc
```

Последовательное рассмотрение файлов языка Python *.py в настоящий момент не очень целесообразно и на первом этапе может быть опущено. Однако для полноты прокомментируем их без детального рассмотрения.

- db.sqlite3 - файл встроенной базы данных, которые использует ваш сайт;
- manage.py –утилита командной строки для выполнения команд Django. Автоматически создается в каждом проекте Django;
- settings.py - - файл настроек, содержащий несколько разделов, сформированных Django для удобства пользователя и включающий некоторые стандартные

функции, присущие обычным web – приложениям. В сайтах пользователя корректируется разработчиком по необходимости;

- utils.py – главный управляющий файл с которого начинается работа любого Django приложения. В некотором смысле это оглавление сайта;
- wsgi.py – WSGI (англ. Web Server Gateway Interface)-совместимые веб-серверы для обслуживания проекта. WSGI — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером;
- __init__.py – показывает Django, что данная папка является пакетом (группой модулей) языка Питон. Создается автоматически. этот файл необходим для того, чтобы Python рассматривал папку **myfirst** как пакет (группу Python-модулей). Файл пустой и добавлять в него, как правило, ничего не требуется;
- файлы в папке **__pycache__** - вспомогательные файлы Python, которые создаются автоматически в Django. Они содержат битовый код, который исполняется интерпретатором Python. Код выполняется на виртуальной машине Python.

Разметки всех текстовых файлов с расширением ***.py** осуществляются по правилам языка Phyton.

Среди перечисленных файлов требуется обратить внимание на файл настроек settings.py. В данном текстовом файле Django сформировал последовательность управляющих инструкций по правилам языка Python, которые сгруппированы в несколько разделов. Если открыть этот файл любым текстовым редактором, то можно увидеть наличие закомментированных строк (начинающихся с #) и строк кода Python. В него можно сразу внести изменения, связанные с региональными настройками, например временем и языком. Изменим для примера стандартную временную настройку, так как Вы находитесь в Европейской части России и хотите, чтобы Django при вызове функций, работающих со временем, устанавливал московское время найдите внизу файла **settings.py** строку вида

…

TIME_ZONE = 'UTC'

…

и замените данную строку (остальной текст не меняйте -отмечено многоточиями) на

…

TIME_ZONE = **'Europe/Moscow'**

…

Если найдете раздел INSTALLED_APPS = в верхней части файла, то увидите, что в стандартном файле настроек Django уже установлено шесть стандартных приложений (app).

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

Кроме того, обратите внимание на строку в файле settings.py

ROOT_URLCONF = 'myfirst.urls'

которую Django сформировал автоматически и в которую попало заданное вами наименование проекта. Это фиксация корневого urls.py файла.

Внесите указанные выше изменения, сохраните и закройте файл. При закрытии измененного файла сервер Django автоматически перезагрузится с учетом изменений в settings.py. В окне командной строки отобразится нечто похожее (что в настоящий момент не имеет существенного значения)

```
E:\NewEx\blog\myfirst\settings.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
You have 17 unapplied migration(s). Your project may not
work properly until you apply the migrations for app(s):
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 24, 2019 - 17:27:08
Django version 2.2.6, using settings 'myfirst.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Очевидно, что представленный набор манипуляций с Django не содержит пока никакой полезной для вас информации, которую вы хотели бы отображать на своем сайте. Пример последовательности создания своего простого проекта продемонстрирован ниже на простейшем web сайте, который исполняет функции интернет блога.

2. Создание простейшего блога.

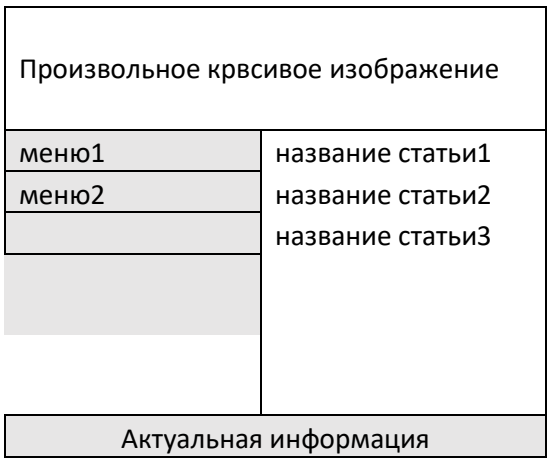
2.1. Каркас для реализации проекта

Блог (англ. blog)— интернет-журнал событий или интернет-дневник. Блог- это веб-сайт, содержимое которого включает добавляемые автором блога текстовые статьи (или иные записи), изображения и мультимедиа. Обычно сайт блога разрешает посетителями сайта публикацию комментариев, например, к текстам статей и т.п.

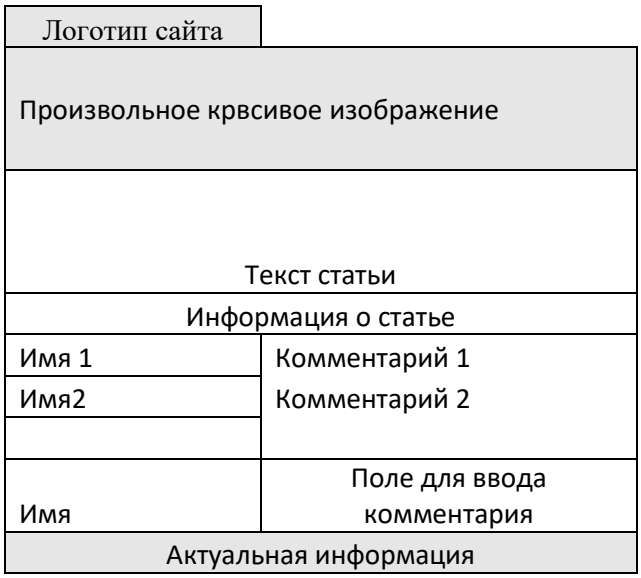
Чтобы конкретизировать пример создадим сайт, который содержит набор статей, подготовленных владельцем блога. Перечень статей расширяется только владельцем блога путем добавления новых статей. При стороннем вызове сайта должен отображаться список названий статей. Так как число статей может быть большим, то разумно на сайт выводить список только последних статей, ранжированный по дате их создания на определенную, заданную временную глубину. Число выводимых названий статей определяется владельцем блога. Посетитель сайта может выбирать статью из списка, читать статью и при желании оставлять свой комментарий к этой статье. При обращении к статье посетитель видит все комментарии с реальными или вымышленными именами посетителей, датами создания комментарий написанных к этой статье предыдущими посетителями. Конечно, дизайн сайта может быть произвольный с использованием стандартных средств языка html и с использованием средств языка css.

Например, структура окна первой страницы сайта может быть такой (на самом деле структуру и дизайн выберет владелец сайта)

Логотип сайта



Соответственно при выборе статьи из списка структура отображаемой страницы может быть следующей



Для реализации такого проекта необходимо создать специальное приложение. В общем случае каждый Django - проект состоит из набора (или одного) приложений. В Django приложение именуется **app (**сокращение от **application**). Как следует из сформулированной задачи основным объектом сайта является объект «статья». И таких статей может быть много. Но к каждой статье привязан текстовый объект «комментарий». К одной статье может быть привязано много комментарий. Другими словами, должна быть использована структурированная база данных, содержащих и статьи, и комментарии, и идентификаторы авторов, и даты создания статей и комментарий к ним и т.п. Кроме того, имеется владелец блога, который имеет уникальное право создавать статьи, удалять статьи, размещать статьи иных авторов, регулировать их параметры и вообще администрировать сайт. И есть посетитель с ограниченными правами читать статьи, по желанию написать и оставлять комментарий, который должен быть доступен иным пользователям, обращающимся к сайту.

Чтобы сформировать такой WEB сайт с использованием Django, создайте новое приложение, например, с именем **articles.** Это и будет приложение, которое включит в

себя выполнение указанных выше возможностей взаимодействия с выбранной статьей. (Если в данный момент вы находитесь в режиме работы сервера разработки необходимо командой Ctrl C ^C или CTRL+ Pause выйти из данного режима). Для создания приложения с именем **articles** (имя задает пользователь) необходимо выполнить в командной строке следующую команду (находясь внутри папки **blog**):

(env_ex) E:\NewEx\blog>**py manage.py startapp articles**

В результате в папке **blog** Django создаст дополнительную папку **articles**. При этом в папке **articles** автоматически создаются заготовки файлов admin.py, apps.py, models.py, test.py, views,py, __init.py__, а также папка **migrations** с файлом __init__.py . Структура и содержание ранее созданных папок (**myfirst, pycache**) не изменится. Структура приведена ниже (жирным шрифтом выделены названия папок, обычным шрифтом название файлов)

E:\NewEx\blog
- db.sqlite3
- manage.py
- **articles**
  - admin.py
  - apps.py
  - models.py
  - tests.py
  - views.py
  - __init__.py
    - **migrations**
      - __init__.py
- **myfirst**
  - settings.py
  - urls.py
  - wsgi.py
  - __init__.py
  - **__pycache__**
    - settings.cpython-37.pyc
    - urls.cpython-37.pyc
    - wsgi.cpython-37.pyc
    - __init__.cpython-37.pyc

Вновь образованные средой Django файлы в папке E:\NewEx\blog**\articles** являются стандартным набором файлов, автоматически создающихся средой Django, для любого вновь созданного приложения пользователем. При этом реальное содержание этих автоматически созданных файлов для пользователя не имеет значения

- admin.py - файл регистрации приложения в модуле. Сейчас в этом файле только одна стандартная инструкция from django.contrib import admin (эта инструкция осуществляет загрузку пакета admin – администрирование сайта)
- apps.py –файл связанный с конфигурированием классов. На данном этапе содержит загрузку пакета AppConfig и две строки кода class ArticlesConfig(AppConfig): name = 'articles'
- models.py – один из трех основных файлов, в котором определяются классы приложения (здесь должно находиться описание модели базы данных).

Сейчас содержит только одну инструкцию from django.db import models (загрузка пакета models)

- tests.py – необязательный специализированный файл, который используется для тестирования приложения. Сейчас содержит одну инструкцию from django.test import TestCase (загрузка пакета TestCase)
- views.py –один из трех основных файлов, в котором записано как представлять данное приложение на сайте. На данном этапе содержит одну строку кода from django.shortcuts import render (загрузка пакета для представления, исполнения содержания файла)
- __init__.py – стандартное объявление о том, что данная папка есть пакет (набор модулей) языка Питон

Немного ниже в этой папке будет создан еще один файл, который нужно создавать самостоятельно - файл urls.py - третий из группы трех (models.py, views.py, urls.py) важнейших файлов, имеющихся во всех приложениях.

Образованная папка E:\NewEx\blog\articles\ **migrations -** это папка в которой Django хранит изменения в базе данных и осуществляет согласование данных с используемой базой.

Если вы хотите только поэкспериментировать с Django, то можно пропустить вопросы, связанные с использованием и настройкой баз данных. Включенная в систему Django база данных будет достаточна для начальных приложений. Но следует отметить, что Django поддерживает четыре СУБД (Система Управления Базой Данных):

- PostgreSQL (http://www.postgresql.org/)
- SQLite 3 (http://www.sqlite.org/)
- MySQL (http://www.mysql.com/)
- Oracle (http://www.oracle.com/)

Все они одинаково хорошо работают с ядром среды Django.
ВАМ НАДО САМОСТОЯТЕЛЬНО ПОЗНАКОМИТЬСЯ С ЭТИМИ БАЗАМИ ДАННЫХ

Запустите сервер (env_ex) E:\NewEx\blog>**py manage.py runserver (**если он не запущен**). Укажите** в браузере системы http://127.0.0.1:8000/ - браузер отобразит начальный стандартный вид начального Django сайта, который сгенерирован автоматически. Никаких изменений Вы не наблюдаете, так как Вы еще ничего нового не сообщили для среды разработки.

Выйдите из сервера разработки командой Ctrl+Pause

Представленная выше замысловатая процедура создания основного каркаса структуры файлов Django может отпугнуть проявляющего нетерпение в желании увидеть результат, но является необходимой по требованию данного фрэйм ворка (птичья терминология присущая программистскому сообществу, даже говорящему не на английском языке). В принципе frame work (это рабочая структура для выполнения работы, рабочая канва, рабочая основа, чего-то). В данном случае – это структура для разработки проекта в среде Django.

По процедурам среды разработки Django утверждается, что дирижером процесса является файл **urls.py**, расположенный в blog\myfirst**\urls.py.** Поскольку в вашем случае создана новая заготовка для приложения **articles**, нужно внести в этот управляющий файл изменения, которые “объяснят” Django где находится файл, управляющий этим новым приложением. Для этого перейдите в папку проекта **blog\myfirst** и отредактируйте файл **urls.py.** Приведенные в файле комментарии или удалите, или оставьте на память – они не играют роли и служат просто минимальной подсказкой (файл сгенерирован Django). Исправьте содержание файла до следующего вида (необходимые исправления выделены жирным шрифтом) и сохраните после исправления: (серым фоном выделен путь к файлу в данном примере)

(env_ex) E:\NewEx\blog\myfirst\urls.py

1. from django.contrib import admin
2. from django.urls import path, **include**

3. urlpatterns = [
4. **….path('', include('articles.urls')),**
5. ….path('admin/', admin.site.urls),
6.]

Что сделано в результате? Django указано, что путь к url.py- файлу, который будет управлять приложением **articles** начинается от базовой директории (которая определена в **settings.py** файле о чем Django позаботился для Вас). Можно просто это увидеть в этом **settings.py** файле, если Вы его откроете любым текстовым редактором. Этот раздел выглядит следующим образом:

Build paths inside the project like this: os.path.join(BASE_DIR, ...)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

Ничего менять не требуется. Просто обратите на это внимание.

Таким образом в blog\myfirst\urls.py добавлен путь к **'articles.urls'**

При этом в важной функции **path**('', include('articles.urls')), первый аргумент ‘’ обозначает базовую директорию к которой Django сам прибавит /articles и в этой папке найдет искомый файл **urls.py**, который будет управлять действием приложения **articles**.

Если в проекте будет несколько **apps** (приложений со своими именами), то в каждом из них будет свой urls.py файл, путь к которому должен быть указан в основном (корневом) urls.py файле. То, что эти файлы имеют одинаковые наименования не мешает при указании пути к файлу.

Теперь, если Вы откроете папку **articles**, то обнаружите, что в ней нет файла **urls.py**. Его должен создать разработчик проекта. Если в проекте много созданных новых приложений, в каждом из них потребуется создавать свои **urls.py** файлы. Перейдите в папку blog\articles\ и создайте в ней новый файл с именем **urls.py** (имя менять нельзя) для данного приложения **article.** Этот URL файл будет отвечать на вопрос как работать с данным приложением **article.** Например, запишем в этот файл такие команды (в соответствии с правилами структурирования файлов Python)

(env_ex) E:\NewEx\blog\articles\

1. from django.urls import path

```
2. from . import views

3. urlpatterns = [
4. ….path('',  views.index, name = 'index'),
5. ]
```

Набор команд означает, что Django должен импортировать функцию **path** и способ просмотра файла **views.py**. Способ просмотра задается какой-то функцией или группой функций. В каждом приложении есть свой файл **views.py**, который объясняет, что конкретно показывать в браузере при вызове **views.py** приложения. Следовательно, именно это нужно указать в файле blog\articles**views.py.** Например, вы хотите, чтобы этот файл выводил на экран сообщение Hellow, world! В этом случае откройте файл blog\articles**views.py** и отредактируйте его следующим образом (все что было в нем записано ранее можно или удалить, или закомментировать):

(env_ex) E:\NewEx\blog\articles**views.py**

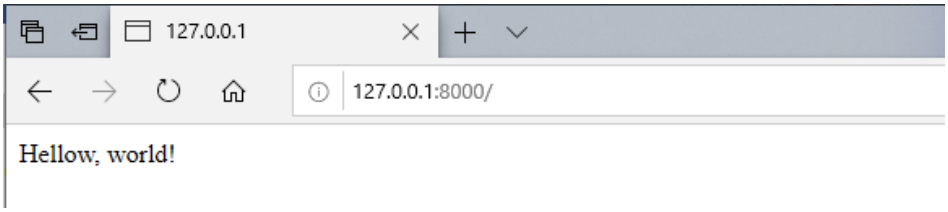
```
1. from django.http import HttpResponse

2. def index(request):
3. ….return HttpResponse("Hellow, world!")
```

Представленный код означает, что в файл вписана функция index аргумент которой служебное слово request (англ. запрос) и эта функция выводит текст, заключенный в двойные кавычки. Это означает, что при обращении к серверу на экране браузера отобразится текст, который указан в этих двойных кавычках. Запустите сервер разработки Django командой в командной строке

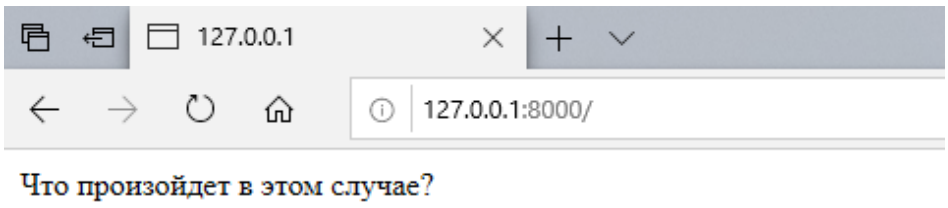
(env_ex) E:\NewEx\blog>**py manage.py runserver**

В строке браузера наберите http://127.0.0.1:8000/

Браузер отобразит **Hellow, world!** Приблизительно следующим образом



Не выходя из сервера разработки откройте любым текстовым редактором файл views.py (env_ex) E:\NewEx\blog\articles**views.py** и измените текст в кавычках на произвольный, например, на ”Что произойдет в этом случае?”. Закройте файл – сервер разработки автоматически обновится. Что отобразится в окне командной строки (смотреть за этим не обязательно). Теперь обновите строку браузера (просто нажмите ввод в заданной вами ранее строке) http://127.0.0.1:8000/ Браузер выведет сообщение:



Поэкспериментируйте с заменой текста (который можно разметить по правилам html языка), чтобы привыкнуть к реакции сервера разработки на изменения в файле. Если это не требуется, то добавьте в urls.py папки **articles** еще один путь (добавленный текст выделен жирно)

(env_ex) E:\NewEx \blog\ myfirst \apps\articles**urls.py**

1. from django.urls import path
2. from . import views
3. urlpatterns = [
4. ….path('', views.index, name = 'index'),
5. **….path('test/', views.test, name= 'test'),**
6.]

При этом сервер разработки в окне командной строки сообщит, что нет атрибута ‘test’ в файле views.py (папки **articles**). В окне командной строки в конце появится сообщение

```
File "<frozen importlib._bootstrap>", line 219, in
_call_with_frames_removed
File "E:\NewEx\blog\articles\urls.py", line 6, in <module>
path('test/', views.test, name= 'test'),
AttributeError: module 'articles.views' has no attribute
'test'
```

Это связано с тем, что добавленная функция **path('test/', views.test, name= 'test'),** требует вызов функции **test** из файла **views.py** при этом содержит базовый путь ‘’ и к нему прибавляется test/ (папки **articles**), однако этой функции в файле **views.py** пока нет. Не выходя из сервера разработки добавьте такой вывод в файле **views.py** (articles). Добавление выделено жирным шрифтом:

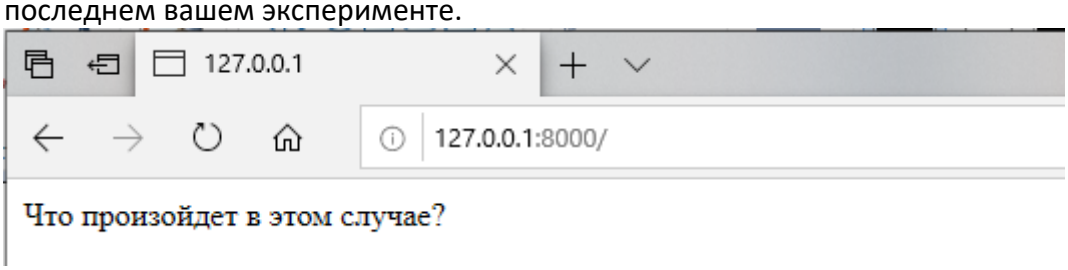
(env_ex) E:\NewEx \blog\ myfirst \apps\articles\views.py

1. from django.http import HttpResponse
2. def index(request):
3. ….return HttpResponse("Hellow, world!")
4. **def test(request):**
5. **….return HttpResponse("Тестовая Страница должна содержать код для тестирования")**

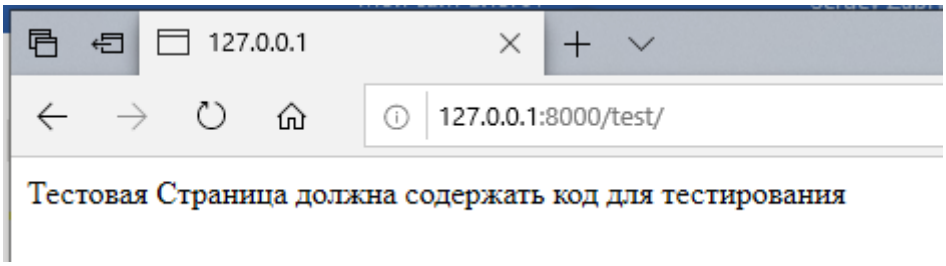
Сервер разработки автоматически обновится (приблизительный вид):

```
You have 17 unapplied migration(s). Your project may not
work properly until you apply the migrations for app(s):
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 25, 2019 - 17:45:16
Django version 2.2.6, using settings 'myfirst.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Теперь, если в браузере вызвать http://127.0.0.1:8000/ то браузер покажет: следующий текст: **Что произойдет в этом случае?** Это в том случае, что вы не экспериментировали с выводом текста. В противном случае будет показана фраза, которую вы ввели при последнем вашем эксперименте.



Если в строке браузера теперь добавить /test/, т.е. вызвать адрес http://127.0.0.1:8000/test/ то на странице отобразится такой текст **Тестовая Страница должна содержать код для тестирования** (тот текст, который вы вписали выше в функции **test**(**request**))



Для примера использования языка html замените в файле views.py разметку выводимого текста (выделено жирным шрифтом)

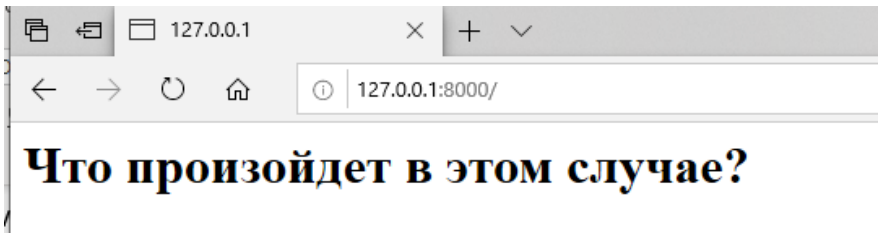
(env_ex) E:\NewEx \blog\myfirst\articles\views.py

1. from django.http import HttpResponse

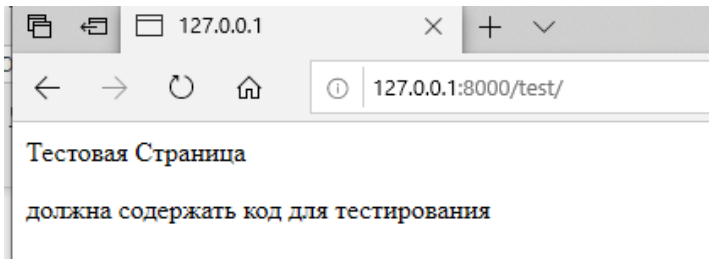
2. def index(request):
3. ….return HttpResponse**("<h1>**Что произойдет в этом случае? **</h1>**")

4. def test(request):
5. ….return HttpResponse("Тестовая Страница **<p>** должна содержать код для тестирования")

Вызовы браузера по адресам http://127.0.0.1:8000/ и http://127.0.0.1:8000/test/ соответственно, приведут к выводу представлений приблизительно так



И



Рассмотренные примеры демонстрируют вывод так называемых статических изображений. Очевидно, что если оглянуться на проделанную работу, то соотношение затраченные усилия - результат выглядят довольно непривлекательно, так как простой html в несколько строчек решает задачу вывода статического текста на экран браузера без всякого труда. Однако все еще не продемонстрированы важные возможности среды разработки Django, такие как администрирование сайта, определение моделей, формирование шаблонов и т.п. Последовательное рассмотрение этих возможностей не целесообразно на данном этапе.

После того как разберётесь детально со всем эти текстом и указанными выше базами данных сообщите. ПОКА ЭТО ВТОРОЕ ЗАДАНИЕ.

========== 001==========