Лекция 15

Разработка Web приложений

Web-сервер

- Web-сервер это программа, принимающая сетевые запросы по протоколу HTTP и отвечающая на них передачей данных в рамках протокола HTTP. Чаще всего это HTML страницы, но в принципе, это могут быть данные любого типа.
- Запрос формируется в виде строки URL, например:

```
http://www.site.org/support/new/item.html
```

- URL содержит:
  - тип протокола, http://, https://, возможны и другие типы
  - название сайта www.site.org, транслируемое в IP адрес
  - директория файла web-страницы /support/new/
  - имя файла web-страницы item.html
- Название сайта, директория и имя файла web-страницы это всего лишь имена, отображаемые в реальные ресурсы средствами Web-сервера

HTTP и HTML

- HTTP это протокол обмена данными между Web-сервером и браузером (web-клиентом). Протокол это *правила диалога* между клиентом и сервером.
- HTTP это *текстовый протокол*, в рамках протокола клиент и сервер обмениваются сообщениями
- Сообщение он состоит из текстового заголовка (блока запроса или ответа) и необязательного блока данных. Блок данных отделяется от заголовка пустой строкой; блок данных может быть бинарным.
- В блоке данных может передаваться любая информация, в частности текст, оформленный в виде HTML
- HTML это *язык представления информации*, он определяет как информация должна быть отображена на экране
- HTML может быть использован как контейнер для передачи произвольной информации, в частности программ на языке Javascript

Web framework - ядро сервера приложений

- *Сервер приложений* это сервер выполняющий программу по запросу пользователя и возвращающий результат ее работы для отображения в браузере (web-клиенте)
- CGI - Common Gateway Interface - первый интерфейс исполнения программ на Web-сервере ставший началом web-технологий
- Наиболее популярные Web frameworks
  - Drupal (PHP)
  - Django (Python)
  - Flask (Python)
  - Ruby on Rails (Ruby)
- Web-сервер совмещенный с сервером приложений
  - Apache Tomcat (java)
- Минимальный Web framework содержащий Web-сервер
  - bottle - представляет собой один файл на Питоне

CGI - Common Gateway Interface

- На Web-сервере создается директория с программами, как правило cgi-bin. *Программы могут быть написаны на любом языке*.
- При запросе к Web-серверу вида

  ```
  http://www.site.org/cgi-bin/progname
  ```

  Web-сервер исполняет программу *progname* и то, что программа выводит в стандартный вывод (stdout) становится web-страницей, которую видит пользователь
- Недостатки CGI:
  - низкая скорость исполнения
  - потенциальная брешь в безопасности системы
  - необходимость генерации полноценной html страницы
- Достоинства CGI:
  - простота

Строки заголовка HTTP протокола

- Строки заголовка HTTP протокола (HTTP headers) позволяют согласовать возможности Web-сервера и Web-браузера, например поддержку сжатия данных при передаче
- Главное назначение строк заголовка это информирование клиента о том, какого рода данные будут ему переданы
- Как следствие, программа вызываемая через CGI должна вывести в стандартный вывод одну или несколько строк HTTP заголовка, пустую строку и лишь затем HTML страницу:

```
Content-Type: text/html; charset=UTF-8

<html><body>
<p>First paragraph</p>
<p>Second paragraph</p>
</body></html>
```

- В результате на Web-странице будет отображено:

```
First paragraph
Second paragraph
```

Web framework

- Web framework это web сервер, который работает как интерактивное приложение, у которого браузер играет роль графического интерфейса пользователя
- Web framework содержит библиотеку подпрограмм, которые исполняются при получении запроса к Web серверу
- Web framework взаимодействует с Web-сервером через специальный интерфейс, для Питона используется интерфейс WSGI. Интерфейс загружает программы и библиотеки Питона в память и позволяет серверу вызывать отдельные функции.
- Большинство Web frameworks имеют встроенный Web-сервер, который используется для отладки приложений. Встроенный Web-сервер имеет невысокую производительность и часто может обслуживать не более одного клиента одновременно
- Для информационного наполнения Web framework использует систему управления базами данных (СУБД)

Django

- Django наиболее популярный Web framework написанный на Питоне

  `https://www.djangoproject.com/`

- Преимущества Django:
  - проект хорошо документирован
  - проект находится в постоянном развитии
  - имеет большое сообщество способное оказать поддержку
  - представляет собой готовый каркас web приложения нуждающийся лишь в наполнении функционалом
  - большое количество дополнительных модулей и приложений, написанных сторонними разработчиками
- Версия Django 3.0 вышла 2 декабря 2019 г.
  - в версии 3.0 введен новый интерфейс ASGI поддерживающий асинхронный вызов процедур

Установка Django

- С точки зрения Питона Django это модуль, импортируемый инструкцией

```
import django
```

- Для работы с Django требуется:
  - Web-сервер, как правило используют Apache2 или nginx
  - Модуль связи Web-сервера с Питоном, как правило WSGI
  - Сервер СУБД
  - Модуль связи СУБД с Питоном, есть возможность работы со всеми популярными СУБД
- Для тестирования можно использовать:
  - Web-сервер входящий в состав Django. Сервер написан на Питоне и специального модуля связи не требуется.
  - Встраиваемую СУБД-библиотеку sqlite3. Интерфейс входит в стандартную библиотеку Питона, данные хранятся в файле.

Виртуальная среда Питона

- Работа с Django требует установки модулей Питона, которые могут конфликтовать с модулями из базовой конфигурации. В таких случаях создается виртуальная среда и работа ведется в ней.
- Создание виртуальная среды в директории /home/user/projenv :

```
pyvenv /home/user/projenv          # устаревший вариант
python3 -m venv /home/user/projenv # рекомендованный вариант
```

- Виртуальная среда это директория, в которую будут устанавливаться модули Питона. Эти модули будут иметь приоритет над модулями из базовой конфигурации.
- Вход в виртуальную среду (Linux, bash):

```
source /home/user/projenv/bin/activate
```

- Выход из виртуальной среды (Linux, bash):

```
deactivate
```

Пакеты в Питоне

- *Пакет* (package) это агрегатор нескольких модулей в единую конструкцию
- Пакет реализован как директория, содержащая файл `__init__.py` Имя директории становится именем пакета. Файл `__init__.py` может содержать код инициализации пакета или быть пустым. В частности в этом файле переменная `__all__` определяет видимость отдельных модулей и находящихся в них объектов.
- Файлы с расширением `.py` в той же директории становятся модулями пакета
- Для импорта модуля из пакета используется точечная нотация:

```
import имя_пакета.имя_модуля
```

- Наряду с модулями пакет может содержать пакеты.
  Таким образом выстраивается древовидная иерархия модулей.

Начало работы

- Вход в виртуальную среду и установка модуля django

```
source /home/user/projenv/bin/activate
pip install django
```

- Создание и вход в рабочую директорию проекта

```
mkdir /home/user/djp && cd /home/user/djp
```

- Создание проекта

```
django-admin startproject sample
```

  будет создана директория sample и следующие файлы в ней:

```
sample/manage.py           # Утилита управления проектом
sample/sample/urls.py      # URL-диспетчер
sample/sample/wsgi.py      # Главный модуль исполняемый на Web-сервере
sample/sample/settings.py  # Файл конфигурации проекта
sample/sample/__init__.py  # Файл инициализации пакета
```

- С точки зрения Питона проект является пакетом

Создание приложения

- В рамках одного проекта может быть создано несколько приложений
- Приложение создается командой

```
python3 manage.py startapp mainapp
```

будет создана директория mainapp и следующие файлы в ней:

```
mainapp/apps.py     # Файл конфигурации приложения
mainapp/models.py   # Файл реализации моделей
mainapp/admin.py    # Файл регистрации моделей
mainapp/views.py    # Файл представлений (view)
mainapp/tests.py    # Файл юнит-тестов
mainapp/__init__.py # Файл инициализации пакета
```

Запуск внутреннего сервера

- Web-сервер включенный в состав Django запускается командой

  ```
  python3 sample/manage.py runserver
  ```

- После запуска внутреннего сервера сайт проекта доступен по адресу:

  ```
  http://127.0.0.1:8000/
  ```

  По умолчанию сайт проекта можно увидеть только в браузере, запущенном на локальном компьютере
- Для обеспечения доступа к сайту в сети следует использовать команду

  ```
  python3 sample/manage.py runserver 0:8000
  ```

- Если при выполнении запроса пользователя происходит ошибка, в ответе сервера формируется web-страница, содержащая сообщение об ошибке

Архитектура MTV

- В противопоставление известной архитектуре MVC (Model-View-Controller) Django описывает свою архитектуру как MTV: *Model-Template-View*
- Модель (model) - это описание данных, хранящихся в СУБД, и реализация методов взаимодействия с ними
- Шаблон (template) - постоянная часть web-страницы, в отдельные поля которой встраиваются данные, полученные в результате исполнения модели
- Представление (view) - это набор callback-функций, каждая из которых вызывается как реакция на определенный URL, запрошенный клиентом. Соответствие между URL и callback-функцией определяется URL-диспетчером.

Обработка запроса пользователя

- Web-сервер получает URL-запрос от пользователя
- Web-сервер через модуль WSGI вызывает Django
- URL-запрос разбирается в URL-диспетчере, находится соответствующая функция в модуле view. Функция вызывается с параметрами, выделенными из URL-запроса
- Функция модуля view извлекает или модифицирует данные используя модели, связывающие Django с СУБД
- Результат работы функции из модуля view применяется к шаблону, в результате чего создается web-страница
- Web-страница встраивается в объект HttpResponse, блок ответа по HTTP протоколу, и передается Web-серверу
- Web-сервер возвращает пользователю ответ в виде блока HTTP протокола из которого браузер пользователя извлекает HTML страницу

Web-приложение это программа, управляемая событиями

URL-диспетчер

- URL-диспетчер определяет список *urlpatterns* в котором определяются соответствия между строками URL и функциями, которые должны быть вызваны, как реакция на них

```
from django.urls import path
from . import views

urlpatterns = [
  path('', views.index),
  path('/<int:run>/', views.get_data),
  path('/<int:run>/<int:probe>/', views.get_data),
]
```

- В результате будут вызваны следующее функции:

```
http://127.0.0.1:8000/    # => views.index(request)
http://127.0.0.1:8000/5   # => views.get_data(request, run=5)
http://127.0.0.1:8000/5/2 # => views.get_data(request, run=5, probe=2)
```

Параметр *request* содержит информацию о запросе web-клиента

Представление

- Представления могут быть оформлены в виде набора функций (views) или методов класса (class-based views)
- Представление на основе класса может наследовать от предопределенных в Django типовых представлений, или представлений, созданных пользователем
- Пример функции-представления:

```
from django.http import HttpResponse

def get_data(request, run, probe=None):
  if probe == None:
    html_page = get_all_run_data(run)
  else:
    html_page = get_run_data_for_probe(run, probe)
  return HttpResponse(html_page)
```

- Если информация хранится в базе данных, функция-представление обращается к методам классов моделей, определенных в файле models.py

Шаблон

- Шаблон это текстовый файл с полями, в которые могут вписываться имена переменных или инструкции управления
- Поля переменных ограничиваются парой из двух фигурных скобок {{ }}
  - при обработке шаблона вместо поля переменной подставляется ее значение
- Поля инструкций ограничиваются парой скобок {% %}
  - при обработке шаблона инструкции исполняются

Условия и циклы в шаблонах

- Условие, инструкция if

```
Result is
{% if condition_1 %}
    {{ var1 }}
{% elif condition_2 %}
    {{ var2 }}
{% endif %}
```

  В условиях могут быть использованы логические операторы, операторы сравнения и оператор in

- Цикл, инструкция for

```
<ul>
{% for item in item_list %}
    <li>{{ item.name }}</li>
{% empty %}
    No items found
{% endfor %}
</ul>
```

Другие инструкции в шаблонах

- autoescape on/off - включает/выключает кодирование специальных символов так, что бы они могли отображаться на HTML странице
- comment - комментарий
- cycle - воспроизводит один один из своих аргументов при каждом обращении, аналогичен итератору
- firstof - воспроизводит первый из своих аргументов, значение которого не приводится к False
- include - включает файл в шаблон
- extends - данный шаблон расширяет другой шаблон, аналогия наследования
- filter - текст внутри шаблона обрабатывается указанным фильтром
- spaceless - удаляет пробелы между тегами HTML

Пример шаблона

- Простой пример шаблона с вставкой значений переменных и инструкцией цикла

```
<html><head>
<title>System Status by time</title>
</head><body>
<h1>{{ page_header }}</h1>
<article>
<ul>
{% for status in status_list %}
  <li>
    Temperature {{ status.temperature }} °C,
    Current {{ status.current }} mA
  </li>
{% endfor %}
</ul>
</article>
<pre>{{ system_info }}</pre>
</body></html>
```

Модель

- Для хранения и манипулирования данными Django использует базы данных (БД)
- Модели Django, это классы, являющиеся отображением реляционных таблиц БД
- Пример моделей:

```
from django.db import models

class FacilityEvent(models.Model):
    event_time = models.DateTimeField(primary_key=True)
    temperature = models.IntegerField()
    current_ma = models.IntegerField()
    comment = models.CharField(max_length=127)

class ProbeEvent(models.Model):
    event_time = models.DateTimeField(primary_key=True)
    voltage = models.IntegerField()
```

- Класс модели определяет структуру реляционной таблицы

Создание элемента данных

- Класс модели содержит атрибут *objects*, являющийся представлением всех объектов модели или, другими словами, всех строк реляционной таблицы
- Метод *create()* атрибута objects создает новую строку, которая сохраняется в реляционной таблице методом *save()*:

```
event = FacilityEvent.objects.create(
  event_time=datetime.now(),
  temperature=27,
  current_ma=32)
event.save()
```

Чтение элементов данных

- Для выборки данных используется методы атрибута objects
  - метод *all()* выбирает все данные
  - метод *filter()* выбирает данные, удовлетворяющие условию фильтрации
  - результатом выборки является объект класса QuerySet, имеющий свойства списка
- Метод objects.get() возвращает *одну строку таблицы*
  - если условию выборки удовлетворяет более одной строки возникает исключение MultipleObjectsReturned
  - если нет строк, удовлетворяющих условию выборки возникает исключение DoesNotExist
- Пример:

```
result_list = FacilityEvent.objects.get(current_ma=32)
```

Пример исполнится без ошибки при условии, что в таблице есть только одна строка со значением тока 32.

Дополнительные возможности моделей

- Модели позволяют определять для таблиц внешние ключи
- Модели позволяют совершать над таблицами операции реляционной алгебры
- Модель предоставляет интерфейс для исполнения команд SQL

Работа с формами

- Форма это web-страница, содержащая поля для ввода данных и сопроводительный текст
- В Django формы представлены классом Form субмодуля forms
- При создании собственных форм создается класс, наследующий от класса Form:

```
from django import forms

class GetDataForm(forms.Form):
  event_time = forms.DateTimeField(label='Date and Time')
  temperature = forms.IntegerField(label='Temperature')
  current_ma = forms.IntegerField(label='Current mA')
```

Поддержка сессий

- Запрос web-страницы и ее возврат в браузер пользователя это единичный акт, начинающийся установлением TCP соединения и завершающийся его разрывом (для HTTP v.1.1)
- Механизм гарантирующий, что несколько последовательных запросов сделаны одним и тем же человеком (web-клиентом), называется сессией
- Механизм аутентификации требует от пользователя идентифицировать себя с помощью пароля и предоставляет запросам в рамках текущей сессии права, предоставленные пользователю
- Django содержит реализацию сессий и аутентификации, соответствующие классы могут быть использованы при разработке собственных приложений

Интерфейс администратора

- В комплект Django входит несколько предустановленных приложений. Одно из них - интерфейс администратора.
- Для работы с интерфейсом администратора необходимо создать пользователя с административными привилегиями:

```
django-admin createsuperuser
```

- Доступ к интерфейсу администратора осуществляется посредством URL

```
http://127.0.0.1:8000/admin/
```

- Интерфейс администратора позволяет создавать пользователей и работать с моделями

Статические файлы

- Django может работать в режиме традиционного Web-сервера, то есть на запрос с конкретным именем файла возвращать этот файл клиенту
  - так например картинки, отображаемые на web-странице, передаются как статические файлы
- Работу со статическими файлам обеспечивает предустановленное приложение django.contrib.staticfiles
- В сети Internet можно найти ряд готовых приложений для Django, например в каталоге

  `https://djangopackages.org/`

Программирование для Web

дополнительные темы

- HTML5
- CSS
- Javascript
- jQuery
- AJAX
- JSON
- SQL

Литература к лекции

1. https://docs.djangoproject.com/

2. Головатый А., Каплан-Мосс Дж. "Django. Подробное руководство, 2-е издание." - Пер. с англ. СПб., Символ-Плюс, 2010. 560 с,. ил.
ISBN 978-5-93286-187-5