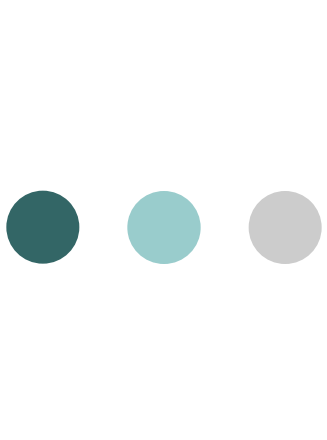


Django

Саяпова Наталья,
222 группа

Django

- Это сильный web-framework, созданный на Python, который следует философии **DRY** (Don’t repeat yourself)
- Он реализован на основе MVC (Model-View-Controller – модель-представление-контроллер).
- Парадигма MVC – идея разделения приложений

Реляционные базы данных

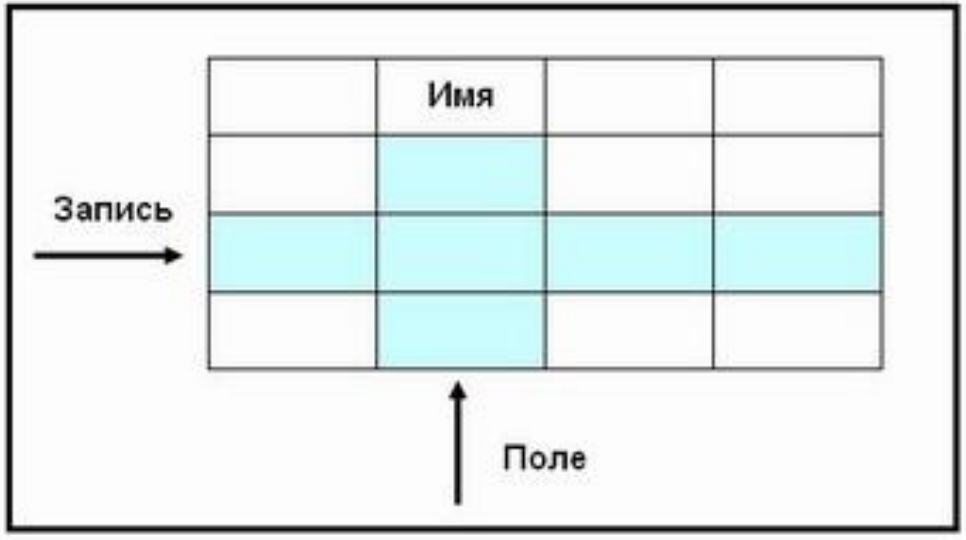
- Базы данных состоят из таблиц, где каждая таблица состоит из строк (например, записей, элементов, объектов) и столбцов (например, атрибутов, полей), напоминая своей организацией электронные таблицы.
- Запросы к базе данных осуществляются с помощью языка SQL(Structured Query language).
- Платформа Django содержит мощную систему ORM, которая классы представляет как таблицы, объект - как отдельные строки внутри этих таблиц, а атрибуты объектов - как столбцы таблиц.

Табличная организация данных

«Игрушки»

Название	Материал	Цвет	Кол-во
мячи	дерево	красный	75
кубики	дерево	голубой	20
куклы	пластмасса	зеленый	34

Объекты	Игрушки (мячи, кубики, куклы)
Запись	Информация об одном объекте (кубики, дерево, голубой)
Поле	Характеристика (атрибут) объекта (резина, дерево, пластмасса)
Имя поля	Название поля, вынесенное в заголовок (материал)



Каждое поле таблицы имеет имя. Например, в таблице «Игрушки» имена полей такие: НАЗВАНИЕ, МАТЕРИАЛ, ЦВЕТ, КОЛИЧЕСТВО.
Одна запись содержит информацию об одном объекте той реальной системы, модель которой представлена в таблице.

Уровни в Django

- ***Model*** — часть, касающаяся доступа к данным; соответствует уровню работы с базой данных;
- ***View*** - часть, касающаяся решения о том, что и как отображать, соответствует представлениям и шаблонам;
- ***Controller*** - часть, которая передает управление некоторому представлению в зависимости от того, что ввел пользователь, реализована самим фреймворком; говорит, какую функцию представления вызывать для данного URL.

Уровни в Django

- ***Model***, уровень доступа к данным. Здесь сосредоточена вся информация о данных: как получить к ним доступ, как осуществлять контроль, каково их поведение, каковы отношения между данными.
- ***Template*** (шаблон), уровень отображения. Здесь принимаются решения, относящиеся к представлению данных: как следует отображать данные на веб-странице или в ином документе.
- ***View***, уровень логики. Здесь расположена логика доступа к модели и выбора подходящего шаблона (или шаблонов). Это мост между моделями и шаблонами.

Взаимодействие уровней

- Запросы HTTP передаются веб-сервером платформе Django, которая принимает их на уровне обработки запросов.
- После этого, исходя из URL, запросы передаются соответствующему представлению, которое выполняет основную часть работы, задействуя при этом модель и/или шаблоны, необходимые для создания ответа.
- Затем выполняется окончательная обработка ответа перед передачей ответа HTTP обратно веб-серверу, который отправляет ответ пользователю.

Слабая связанность

- У каждого компонента веб-приложения, созданного на базе Django, имеется единственное назначение, поэтому его можно изменять независимо от остальных компонентов.
- Есть возможность задействовать ту долю платформы Django, которая требуется, и заменять ее компоненты другими инструментами, которые лучше подходят для решения поставленных задач.

Красивый URL

- В Django невозможно породить конструкции типа: *«index.php?func=user&subfunc=add&PHPSESSIONID=…»*
- Имеется файл, в котором пишется список всех видов URL, которые привязываются к своим обрабатывающим функциям. Причем изменяемые части URL (id и тому подобные) передаются в обработчик в виде обычных параметров функции.

«Питоничность» Django

- Использование краткого, но мощного синтаксиса.
- Должен существовать один — и, желательно, *только* один — очевидный способ сделать это.
- Предпочтение явного перед неявным.
- DRY – надо стараться исключать дублирование уже введенного в систему знания.

Less code

- Django обладает немалым количеством уже написанных базовых вещей:
- ***Сессии***. Достаточно подключить в приложение нужный модуль, и в каждом запросе появится *request.session*, в которую можно класть любые данные, естественно, разные для каждого юзера.
- ***Авторизация***, поддерживающая: регистрацию, авторизацию, систему прав на объекты вашей модели данных, генерацию паролей, рассылку сообщений по e-mail.

Less code

- ***Кеширование***. Для того, чтобы не обращаться в базу каждый раз, когда требуются редко меняющиеся данные, можно закешировать результат.
- ***Административная панель***. Она уже готова к использованию, и разработчику не придется корпеть над ее «обустройством». Нужно лишь указать, какие объекты вы хотите видеть в интерфейсе

Другие возможности Django

- Подключаемая архитектура приложений, которыми можно компоновать целевую информационную систему.
- Система фильтров (англ. *middleware*) для построения дополнительных обработчиков запросов.
- Интернационализация приложений

Установка Django

- Скачать Django можно отсюда: http://www.djangoproject.com/download/
- Django работает с версиями Python 2.3 – 2.6
- В системную переменную Path надо добавить(!) путь до Python
- В командной строке из каталога Django вызываем: setup.py install
- Проверка: в Python Shell

```
>>> import django
>>> django.VERSION
(1, 1, 2, 'final', 0)        #установка прошла успешно
```

Создание проекта

- К системной переменной Path добавляем путь до django-admin.py;
- В домашнем каталоге создаем папку, например, djangocode;
- В командной строке из этого каталога вызываем: django-admin.py startproject mysite
- В папке djangocode появилась папка mysite

Проект

- ***__init__.py***. Пустой файл. Необходим для того, чтобы система рассматривала mysite как модуль Python. Обычно не изменяется.
- ***manage.py***. Позволяет запускать разные команды для администрирования сайта. Изменять не стоит.

Cmd: manage.py help

Проект

- ***urls.py.*** URL для данного проекта. Вначале пуст.
- ***settings.py.*** Файл настроек данного проекта. Здесь указывается, какая база данных используется, сколько хранятся cookies...

Сервер разработки

- Сервер разработки Django — это встроенный упрощенный веб-сервер, которым можно пользоваться в ходе разработки сайта. Он включен в состав Django для того, чтобы можно было быстро создать сайт, не отвлекаясь на настройку полноценного сервера.

CMD: manage.py runserver

Сервер разработки

```
G:\djangocode\mysite>manage.py runserver
Validating models...
0 errors found

Django version 1.1.2, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[29/Nov/2010 22:50:49] "GET / HTTP/1.1" 200 2053
```

Сервер разработки

- По умолчанию команда runserver запускает сервер разработки на порту 8000 и принимает запросы на соединения только с локального компьютера.

  Cmd: manage.py runserver 8080

- Задав также IP-адрес, вы разрешите серверу принимать запросы на соединение с другого компьютера. IP-адрес 0.0.0.0 разрешает серверу прослушивать все сетевые интерфейсы:

  Cmd: manage.py runserver 0.0.0.0:8000

  Теперь любой пользователь в локальной сети сможет увидеть ваш django-сайт, введя в адресной строке своего броузера ваш IP-адрес (например, *http://192.168.1.103:8000/).*

- Чтобы узнать адрес своего компьютера в локальной сети:

  Cmd: ipconfig

Создание приложения

- Cmd: manage.py startapp books
- books/
- __init__.py
- models.py
- views.py

Hello, world!

- В views.py:

```
from django.http import HttpResponse

def hello(request):

    return HttpResponse('Hello, world!')
```

- Необходимо сообщить Django, что при некотором URL должно активироваться представление hello.

Конфигурация URL

- Это оглавление сайта.
- В urls(без комментариев):

```
from django.conf.urls.defaults import *
urlpatterns = patterns('', )
```

- Привязка для представления hello:

```
from django.conf.urls.defaults import *
from mysite.views import hello
urlpatterns = patterns('', ('^hello/$', hello), )
```

Шаблоны

```
<html>
<head><title>Ordering notice</title></head>
<body>

 <h1>Ordering notice</h1>
<p>Dear {{ person_name }},</p>
<p> Thanks for placing an order from {{ company }}. It's scheduled to ship
    on {{ ship_date|date:"F j, Y" }}.</p>
<ul>
{% for item in item_list %}
   <li>{{ item }}</li>
{% endfor %}
</ul>
</body>

</html>
```

Шаблоны

- Информация, которая передается шаблону для отображения, называется *контекстом*. Объект Context похож на словарь. Контекст заполняется либо автоматически(добавление - extra_context), либо самостоятельно(метод render, вспомогательная функция render_response)

Шаблоны

```
>>> from django.template import Context, Template
>>> t = Template('My name is {{ name }}.')
>>> c = Context({'name': 'Stephane'})
>>> t.render(c)
u'My name is Stephane.'
```

Шаблоны

```
from django.template import Template, Context
from django.http import HttpResponse
import datetime
def current_datetime(request):
    now = datetime.datetime.now()
    fp = open('/home/djangouser/templates/mytemplate.html')
    t = Template(fp.read())
    fp.close()
    html = t.render(Context({'current_date': now}))
    return HttpResponse(html)
```

Шаблоны

- Загрузка

В settings.py:

```
TEMPLATE_DIRS = (
    ‘G:/home/django/mysite/templates', )
from django.template.loader import get_template
#...
    t = get_template('current_datetime.html')
#...
```

Шаблоны

- ***Тег Include***

<html> <body>

{% include "includes/nav.html" %}

<h1>{{ title }}</h1>

</body> </html>

- ***Наследование***

Тег {%block%} {% endblock %}

{% extends "base.html" %} – «дочерность»

Модели

```
from django.db import models
class Publisher(models.Model):
    name = models.CharField(max_length=30)
    address = models.CharField(max_length=50)
    website = models.URLField()
class Author(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=40)
    email = models.EmailField()
class Book(models.Model):
    title = models.CharField(max_length=100)
    authors = models.ManyToManyField(Author)   #многие ко многим
    publisher = models.ForeignKey(Publisher)      #один ко многим
    publication_date = models.DateField()
```

Модели

- MIDDLEWARE_CLASSES = (#'django.middleware.common.CommonMiddleware', #'django.contrib.sessions.middleware.SessionMiddleware',
- #'django.contrib.auth.middleware.AuthenticationMiddlewar e',)
- INSTALLED_APPS = (#'django.contrib.auth', #'django.contrib.contenttypes', #'django.contrib.sessions',
- #'django.contrib.sites',
- 'mysite.books',)

Модели

- manage.py validate – корректность задания модели
- manage.py sqlall books – посмотреть, что передано в базу данных
- manage.py syncdb – синхронизация моделей с базой данных

Модели

- >>> from books.models import Publisher
- >>> p1 = Publisher(name='Addison-Wesley', address='75 Arlington Street‘, website='http://www.apress.com/')
- >>> p1.save()
- >>> p2 = Publisher(name="O'Reilly", address='10 Fawcett St.‘, website='http://www.oreilly.com/')
- >>> p2.save()
- >>> publisher_list = Publisher.objects.all()
- >>> publisher_list
- [<Publisher: Publisher object>, <Publisher: Publisher object>]

- Publisher_list = Publisher.objects.filter(name = ‘O’Reilly’)

Модели

- class Author(models.Model):
- first_name = models.CharField(max_length=30)
- last_name = models.CharField(max_length=40)
- email = models.EmailField()
- def __unicode__(self): return '%s %s' % (self.first_name, self.last_name)
- >>> from books.models import Author
- >>> author_list = Author.objects.all()
- >>> author_list [<Author: Mark Twain>]

Почему Django?

- Python
- Бесплатность
- Разделение логики и представления
- Диспетчер URL
- Шаблонизатор
- ORM
- Интерфейс администратора
- Аутентификация и авторизация
- Кэширование
- Удобная интернационализация проектов
- Работа с электронной почтой
- Большое сообщество разработчиков, доступная

Google App Engine

- Google предоставляет свои сервера
- Ограничения:
- Нет доступа на запись в файловую систему сервера. Единственный способ сохранять данные — внутреннее хранилище, нереляционная, высокомасштабируемая база данных.

- Google
- Yandex
- Youtube

Литература

- [1] : Django - Разработка веб-приложений на Python (Джефф Форсье) [2009]
- [2] : Django - Подробное руководство, 2-е издание (Адриан Головатый) (2010)