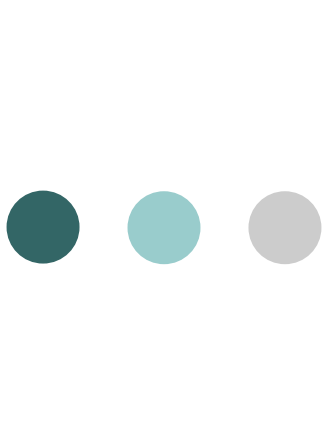


Django

Django

- Бұл DRY (өзіңізді қайталамаңыз) философиясына сәйкес келетін Python-да жасалған күшті веб-фремворк.
- Ол MVC (Model-View-Controller - Model-View-Controller) негізінде жүзеге асырылады.
- MVC парадигмасы - қолданбаларды бөлу идеясы

Реляциялық мәліметтер қоры

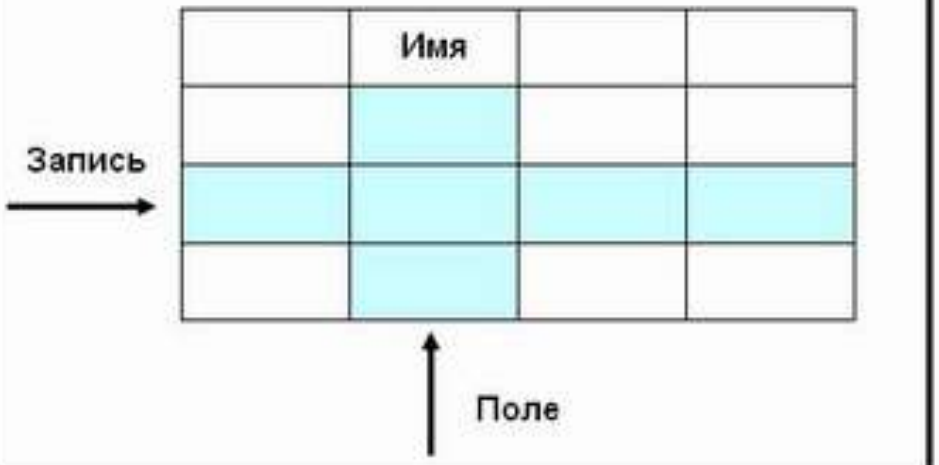
- Мәліметтер қоры кестелерден тұрады, мұнда әрбір кесте жолдардан (мысалы, жазбалар, элементтер, нысандар) және бағандардан (мысалы, атрибуттардан, өрістерден) тұрады, олар ұйымдастыру жағынан электрондық кестелерге ұқсас.
- Деректер базасына сұраныстар құрылымдық сұраныс тілі (SQL) арқылы жасалады.
- Django платформасында сыныптарды кестелер ретінде, нысанды сол кестелердегі жеке жолдар ретінде және нысан атрибуттарын кестелердің бағандары ретінде көрсететін қуатты ORM жүйесі бар.

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)



Әрбір кесте өрісінің атауы бар. Мысалы, «Ойыншықтар» кестесінде өріс атаулары: АТЫ, МАТЕРИАЛ, ТҮС, САНЫ.

Бір жазба осы нақты жүйенің бір объектісі туралы ақпаратты қамтиды, оның моделі кестеде берілген.

Джангодағы деңгейлер

- *Model – деректерге қол жеткізуге қатысты бөлік; мәліметтер қорымен жұмыс деңгейіне сәйкес келеді;*
- *View - нені және қалай көрсету керектігін шешуге арналған бөлік, көріністер мен үлгілерге сәйкес келеді;*
- *Controller - қолданушы енгізген нәрсеге байланысты басқаруды қандай да бір көріністе тасымалдайтын бөлік, фреймворктің өзі жүзеге асырады; берілген URL мекенжайына қандай қарау функциясын шақыру керектігін айтады.*

Джангодағы деңгейлер

- Model, деректерге қол жеткізу деңгейі. Мұнда деректер туралы барлық ақпарат шоғырланған: оған қалай қол жеткізуге болады, бақылауды қалай жүзеге асыруға болады, оның мінез-құлқы қандай, деректер арасындағы байланыстар қандай.
- Template (үлгі), дисплей деңгейі. Бұл жерде презентация туралы шешімдер қабылданады: деректер веб-бетте немесе басқа құжатта қалай көрсетілуі керек.
- View, логикалық деңгей. Модельге қол жеткізу және сәйкес үлгіні (немесе үлгілерді) таңдау логикасы осы жерде орналасқан. Бұл үлгілер мен шаблондар арасындағы көпір.

Деңгейдегі өзара әрекеттесу

- HTTP сұрауларын веб-сервер Django платформасына жібереді, ол оларды сұранысты өңдеу деңгейінде қабылдайды.
- Осыдан кейін URL мекенжайына негізделген сұраулар жауап жасау үшін қажетті үлгіні және/немесе үлгілерді пайдалану кезінде жұмыстың негізгі бөлігін орындайтын сәйкес көрініске жіберіледі.
- Жауаптың соңғы өңдеуі HTTP жауабын веб-серверге қайтару алдында орындалады, ол жауапты пайдаланушыға кері жібереді.

Әлсіз байланыс

- Әрбір Django негізіндегі веб-бағдарлама құрамдас бөлігі бір мақсатқа ие, сондықтан оны басқа компоненттерден тәуелсіз өзгертуге болады.
- Джанго платформасының қажетті бөлігін пайдаланып, оның құрамдастарын тапсырмаға жақсырақ сай келетін басқа құралдармен ауыстыруға болады.

Красивый URL

- Django-да келесідей конструкцияларды жасау мүмкін емес: "index.php?func=user&subfunc=add&PHPSESSIONID=..."
- Өңдеу функцияларына байланысты URL мекенжайларының барлық түрлерінің тізімін қамтитын файл бар. Сонымен қатар, URL мекенжайының айнымалы бөліктері (id және т.б.) өңдеушіге қарапайым функция параметрлері түрінде беріледі.

«Питоничность» Django

- Қысқа, бірақ күшті синтаксисті пайдалану.
- Мұны істеудің бір, ең дұрысы, бір ғана жолы болуы керек.
- Айқынға қарағанда айқынды артықшылық.
- DRY - жүйеге енгізілген білімдердің қайталануын болдырмауға тырысу керек.

Less code

- Джангода қазірдің өзінде жазылған көптеген негізгі материалдар бар:
- Сессия. Қолданбаға қажетті модульді қосу жеткілікті, және әрбір сұрауда сұрау.сеанс пайда болады, оған кез келген деректерді, әрине, әр пайдаланушы үшін әртүрлі қоюға болады.
- Қолдайтын авторизация: тіркеу, авторизация, деректер үлгісінің объектілеріне құқықтар жүйесі, парольді құру, электрондық пошта арқылы хабарламалар жіберу.

Less code

- Кэштеу. Деректерді сирек өзгерту қажет болған сайын дерекқорға қол жеткізбеу үшін нәтижені кэштеуге болады.
- Әкімшілік панель. Ол қазірдің өзінде пайдалануға дайын және әзірлеушіге оның «орналастыруына» көз жүгіртудің қажеті жоқ. Сізге интерфейсте қандай нысандарды көргіңіз келетінін көрсету керек

Другие возможности Django

- Мақсатты ақпараттық жүйені жинауға болатын қосылатын қолданба архитектурасы.
- Қосымша сұраныс өңдеушілерін құруға арналған сүзгі жүйесі (ағылшынша аралық бағдарлама).
- Қолданбаларды интернационалдандыру

Установка Django

- Сіз Django-ны мына жерден жүктей аласыз: http://www.djangoproject.com/download/
- Django Python 2.3 - 2.6 нұсқаларымен жұмыс істейді
- Path жүйесінің айнымалы мәніне Python жолын қосыңыз(!).
- Пәрмен жолында Django каталогынан біз мынаны шақырамыз: setup.py install
- Тексеру: Python Shell ішінде
- >>> импорттау джанго
- >>> django.VERSION
- (1, 1, 2, 'соңғы', 0) #орнату сәтті аяқталды

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 документация.

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)