Лекция 2 марта 2016

Основы веб-разработки (первый семестр)

Примеры питонячих демонов

- https://github.com/sat2707/web/blob/master/tcp_servers/simple.py
- https://github.com/sat2707/web/blob/master/tcp_servers/simple_http.py
- https://github.com/sat2707/web/blob/master/tcp_servers/fork.py
- https://github.com/sat2707/web/blob/master/tcp_servers/prefork.py
- https://github.com/sat2707/web/blob/master/tcp_servers/async.py

Примеры конфы nginx

- https://github.com/sat2707/web/blob/master/tcp_servers/nginx.conf
- https://github.com/sat2707/web/blob/master/tcp_servers/itportal.conf
- https://github.com/sat2707/web/blob/master/tcp_servers/mipt.conf

UPD. Прошу прощения, конкретика по ДЗ и туториал по нему немного задерживаются, будут завтра.
Презентации к первой и второй лекции можно взять тут

- https://s.mail.ru/2JoxRjj2SZbC/web%201.pdf
- https://s.mail.ru/372BnaJVTX47/web2.pdf

UPD2. Домашнее задание.

Ваша задача, как я и говорил на лекции, поставить nginx, gunicorn и django + добиться того, чтобы все эти компоненты смогли работать вместе. Я предполагаю, что вы будет всё ставить себе на ноуты, на которых у вас установлен ubuntu.
В убунту системные пакеты (например, nginx) устанавливаются с помощью пакетного менеджера apt-get. О его устройстве нам особо ничего знать не нужно, мы им будем пользоваться только в разрезе "ой я хочу вот nginx, apt-get поставь мне его пожалуйста" :)
Попробую попутно объяснять, что мы делаем и зачем. Оставшиеся вопросы можете накидывать в комментарии. Начнем..

Ставим django

1) Прежде всего создаем папку под проект (я проект назову допустим stackoverflow, вы можете назвать как вам угодно)

Прямой эфир

Марина Даньшина 14 минут назад
Расписание занятий → Расписание пересдач 0

Екатерина Рогушкова 8 часов назад
Получение значков достижений на портале (ачивки) 2

Ольга Августан Вчера в 18:41
Опросы → Получение значков достижений на портале (ачивки) 2

Ольга Августан Вчера в 13:19
Общие вопросы → Завершение семестра: последние итоговые встречи 0

Ольга Августан Вчера в 13:12
Завершение семестра: итоговые встречи по дисциплинам 10

Марина Даньшина 25 Мая 2016, 14:01
Разработка на Java (первый семестр) → Забыли зонтик 0

Ksenia Sternina 25 Мая 2016, 13:38
Домашнее задание - Юзабилити-тестирование 4

Летяго Владимир 25 Мая 2016, 01:40
Разработка приложений на Android - 1 (второй семестр) → Контрольный рубеж 0

Летяго Владимир 24 Мая 2016, 22:45
Домашнее задание №3 10

Вячеслав Ишутин 24 Мая 2016, 22:31
Разработка на C++ (открытый курс) → Итоговая встреча по дисциплине 0

Вячеслав Ишутин 24 Мая 2016, 22:22
Разработка на C++ (открытый курс) → Итоговое занятие. Презентация курсовых проектов. 0

Александр Агафонов 24 Мая 2016, 19:17
Мастер-класс Ильи Стыценко онлайн «Использование OAuth 2 в приложениях на Django» 1

Марина Даньшина 23 Мая 2016, 20:59
Мероприятия → Мастер-класс Ильи Стыценко онлайн «Использование OAuth 2 в приложениях на Django» 1

Ольга Августан 23 Мая 2016, 17:56
Разработка приложений на iOS - 1 (второй

Связь с разработчиками

Папка создается командой mkdir имя_папки

```
vagrant@precise64:~$ mkdir stackoverflow
vagrant@precise64:~$ ls
stackoverflow
vagrant@precise64:~$ cd stackoverflow/
vagrant@precise64:~/stackoverflow$ pwd
/home/vagrant/stackoverflow
vagrant@precise64:~/stackoverflow$
```

2) Убеждаемся, что у нас есть python и он нужной версии

```
vagrant@precise64:~/stackoverflow$ python -V
Python 2.7.3
vagrant@precise64:~/stackoverflow$
```

3) Ставим утилиту pip. Это python install packages - специальный менеджер пакетов для python, который существенно облегчает установку и сборку различных библиотек для языка python, да и вообще любых утилит, написанных на python (например, django это по факту именно библиотека на python, а gunicorn - это написанный на python сервер).
Итак:

```
vagrant@precise64:~/stackoverflow$ sudo apt-get install python-pip
Reading package lists... Done
...
Setting up python-pip (1.0-1build1) ...
vagrant@precise64:~/stackoverflow$
```

Здесь мы использовали sudo - эта команда позволяет выполнить какие-либо операции от имени администратора (который root). Собственно, когда требуется выполнить какую-либо операцию, требующую прав суперпользователя - используется sudo.

4) Ставим утилиту virtualenv. Это менеджер виртуальных окружений для python. Попробую объяснить, что это означает "нормальным языком". Если просто поставить например django через pip install, то у нас для всей системы будет установлена django последней версии. В то же время, вы вполне можете заниматься разработкой двух проектов, при этом для одного может требоваться django одной версии, а для другого - другой (например, я разрабатываю порядка 6-7 разных проектов, для которых мне одновременно нужны django четырех разных версий, от 1.6 до 1.9). Кроме того, обычно в ходе разработки ставится целая куча сторонних библиотек (штук 20-30 - нормальная ситуация). Таким образом, наша задача - иметь возможность использовать для разных проектов разные версии библиотек python, каждую - со своими библиотеками, так чтобы они не конфликтовали друг с другом. Для этого и используется утилита virtualenv - она дает возможность создать в отдельной папке копию интерпретатора языка python с собстенными настройками и собственным набором библиотек.

семестр) → Завершение семестра 0

Ольга Августан 23 Мая 2016, 17:16
Итоговая встреча по дисциплине 2

Ольга Августан 23 Мая 2016, 17:07
Основы веб-разработки (первый семестр) → Итоговая встреча по дисциплине 0

Ольга Августан 23 Мая 2016, 17:06
Основы мобильной разработки (первый семестр) → Итоговая встреча по дисциплине 2

Ольга Августан 23 Мая 2016, 17:05
Проектирование интерфейсов мобильных приложений (второй семестр) → Итоговая встреча по дисциплине 0

Ольга Августан 23 Мая 2016, 17:02
Проектирование СУБД (второй семестр) → Итоговая встреча по дисциплине 0

Ольга Августан 23 Мая 2016, 16:59
Общие вопросы → Завершение семестра: итоговые встречи по дисциплинам 10

Весь эфир

Блоги

Основы веб-разработки (первый семестр) 26,04
Разработка на C++ (открытый курс) 16,97
Основы мобильной разработки (первый семестр) 14,70
Разработка на Java (первый семестр) 13,58
Общие вопросы 5,71
Разработка приложений на Android - 1 (второй семестр) 5,66
Стажировка 4,52
Проектирование интерфейсов мобильных приложений (второй семестр) 3,42
Разработка приложений на iOS - 1 (второй семестр) 2,29
Проектирование СУБД (второй семестр) 2,26

Так вот. для начала установим virtualenv с помощью pip:

```
vagrant@precise64:~/stackoverflow$ sudo pip install virtualenv
Downloading/unpacking virtualenv
...
Successfully installed virtualenv
Cleaning up...
vagrant@precise64:~/stackoverflow$
```

Пользоваться данной штукой весьма просто - вы просто набиваете virtualenv имя_папки, и в этой папке у вас создается отдельное окружение со своей копией языка python, и вы в любой момент это окружение можете активировать командой source имя_папки/bin/activate и деактивировать командой deactivate. Пробуем:

```
vagrant@precise64:~/stackoverflow$ virtualenv env
New python executable in /home/vagrant/stackoverflow/env/bin/python
Installing setuptools, pip, wheel...done.
vagrant@precise64:~/stackoverflow$
vagrant@precise64:~/stackoverflow$ source env/bin/activate
(env) vagrant@precise64:~/stackoverflow$
(env) vagrant@precise64:~/stackoverflow$ pip install django
Collecting django
...
Successfully installed django-1.9.4
(env) vagrant@precise64:~/stackoverflow$
(env) vagrant@precise64:~/stackoverflow$ deactivate
vagrant@precise64:~/stackoverflow$
```

Мы только что создали окружение в папке env, активировали его и поставили туда django. Потом деактивировали ("вышли из окружения"). Теперь у нас есть собственная версия интерпретатора python в папке env/, и там установлен django версии 1.9.4
Обратите внимание, когда мы внутри виртуального окружения - у нас меняется приглашение сервера (добавляется имя, в нашем случае env).

5) Снова активируем окружение и ставим сервер gunicorn:

```
vagrant@precise64:~/stackoverflow$ source env/bin/activate
(env) vagrant@precise64:~/stackoverflow$ pip install gunicorn
Collecting gunicorn
...
Successfully installed gunicorn-19.4.5
(env) vagrant@precise64:~/stackoverflow$ pip freeze
Django==1.9.4
gunicorn==19.4.5
(env) vagrant@precise64:~/stackoverflow$
```

Обратите внимание, sudo нам внутри виртуального окружения уже не требуется,

поскольку мы ставим питонячьи пакеты не для всех системы, а только для нашего пользователя, так что админские права нам не нужны.
Команда pip freeze показывает список пакетов, установленных в данный момент, вместе с версиями. У нас внутри виртуального окружения установлены только django и gunicorn. Если же выполните pip freeze вне окружения - увидите, что для системы в целом стоит другой набор библиотек.

6) Создаем папку с исходниками проекта, и кладем туда файл с зависимостями:

```
(env) vagrant@precise64:~/stackoverflow$
(env) vagrant@precise64:~/stackoverflow$ mkdir src
(env) vagrant@precise64:~/stackoverflow$ pip freeze > src/requirements.txt
(env) vagrant@precise64:~/stackoverflow$ cat src/requirements.txt
Django==1.9.4
gunicorn==19.4.5
(env) vagrant@precise64:~/stackoverflow$
```

Только что мы выполнили команду pip freeze и перенаправили у нее поток вывода в файл requirements.txt. Таким образом в requirements.txt у нас сейчас лежит описание того, какие библиотеки и каких версий нам нужны для данного проекта. Так что теперь на любом сервере мы можем создать виртуальное окружение, выполнить в нем pip install -p requirements.txt и установить ровно тот набор библиотек, который нам нужен. Обратите внимание, что обычно в requirements.txt складывается не полностью весь выхлоп от pip freeze, а только те библиотеки, которые были установлены создательно. Бывает, что ставишь библиотеку "A", а она для работы требует библиотеку "B", и "B" ставится автоматически. Так вот, библиотека "B" в requirements.txt светиться не должна, то есть смотрим на вывод pip freeze, находим там то, что только что установили и добавляем вместе с версией в requirements.txt.

7) В папке src создадим собственно django-проект, который назовем например application (или как хотите можете его назвать :)

```
(env) vagrant@precise64:~/stackoverflow$ django-admin startproject application src
(env) vagrant@precise64:~/stackoverflow$ ls src
application manage.py requirements.txt
(env) vagrant@precise64:~/stackoverflow$
```

Команда django-admin у нас появилась в виртуальном окружении после установки django. Папка application - это папка проекта, там вы будете вести собственно разработку. Файл manage.py - это файл управления проектом, дальнейшая административная работа с проектом делается через ./manage.py имя_команды

8) Пробуем запустить сервер django, сначала зайдя в папку src

```
(env) vagrant@precise64:~/stackoverflow$ cd src
(env) vagrant@precise64:~/stackoverflow/src$ ./manage.py runserver localhost:8080
Performing system checks...
```

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.

March 06, 2016 - 15:43:26
Django version 1.9.4, using settings 'application.settings'
Starting development server at http://localhost:8080/
Quit the server with CONTROL-C.
^C(env) vagrant@precise64:~/stackoverflow$

Вырубить его можно через Ctrl+C. А пока он запущен - можно зайти на localhost:8080 и посмотреть, что оно действительно "чота делает, чота заработало"

9) Пробуем запустить сервер django через wsgi-сервер gunicorn (убиваем тоже с помощью ctrl+C)

(env) vagrant@precise64:~/stackoverflow/src$ gunicorn --reload -b localhost:8080 application.wsgi:application
[2016-03-06 15:57:08 +0000] [1651] [INFO] Starting gunicorn 19.4.5
[2016-03-06 15:57:08 +0000] [1651] [INFO] Listening at: http://127.0.0.1:8080 (1651)
[2016-03-06 15:57:08 +0000] [1651] [INFO] Using worker: sync
[2016-03-06 15:57:08 +0000] [1656] [INFO] Booting worker with pid: 1656
^C[2016-03-06 15:57:26 +0000] [1651] [INFO] Handling signal: int
[2016-03-06 15:57:26 +0000] [1656] [INFO] Worker exiting (pid: 1656)
[2016-03-06 15:57:26 +0000] [1651] [INFO] Shutting down: Master
(env) vagrant@precise64:~/stackoverflow/src$

параметр --reload заставляет gunicorn самостоятельно перезапускаться, когда вы меняется код в джанго-проекте. Джанговский runserver делает это по-дефолту
параметр -b позволяет указать, какие ip-адреса мы слушаем и на каком порту. В нашем случае - localhost порт 8080
в конце мы указываем путь до callback-функции, которую должен вызывать gunicorn каждый раз, когда ему приходит запрос. мы указали на функцию application в файле application/wsgi.py

10) Пробуем то же самое, но с использованием четырех воркеров (то, что я рассказывал про prefork)

(env) vagrant@precise64:~/stackoverflow/src$ gunicorn --reload -b localhost:8080 application.wsgi:application -w 4
[2016-03-06 15:57:31 +0000] [1662] [INFO] Starting gunicorn 19.4.5
[2016-03-06 15:57:31 +0000] [1662] [INFO] Listening at: http://127.0.0.1:8080 (1662)
[2016-03-06 15:57:31 +0000] [1662] [INFO] Using worker: sync
[2016-03-06 15:57:31 +0000] [1667] [INFO] Booting worker with pid: 1667
[2016-03-06 15:57:31 +0000] [1669] [INFO] Booting worker with pid: 1669

```
[2016-03-06 15:57:31 +0000] [1679] [INFO] Booting worker with pid: 1679
[2016-03-06 15:57:31 +0000] [1681] [INFO] Booting worker with pid: 1681
^C[2016-03-06 15:57:36 +0000] [1662] [INFO] Handling signal: int
[2016-03-06 15:57:36 +0000] [1681] [INFO] Worker exiting (pid: 1681)
[2016-03-06 15:57:36 +0000] [1667] [INFO] Worker exiting (pid: 1667)
[2016-03-06 15:57:36 +0000] [1669] [INFO] Worker exiting (pid: 1669)
[2016-03-06 15:57:36 +0000] [1679] [INFO] Worker exiting (pid: 1679)
[2016-03-06 15:57:36 +0000] [1662] [INFO] Shutting down: Master
(env) vagrant@precise64:~/stackoverflow/src$
```

Так оно будет работать гораздо быстрее

11) Создадим две папки - collected_static и media (первая - для статики, вторая - для файлов, загруженных будущими пользователями). Обе папки у нас впоследствии будут обслуживаться nginx-ом

```
(env) vagrant@precise64:~/stackoverflow/src$ cd ..
(env) vagrant@precise64:~/stackoverflow$ mkdir collected_static media
(env) vagrant@precise64:~/stackoverflow$ ls
collected_static env src media
(env) vagrant@precise64:~/stackoverflow$
```

Объсняю, зачем это нужно. Ну, с папкой media всё должно быть понятно - если пользователь загружает на сервер картинку (например, собственную аватарку), то django будет класть эту картинку в папку media (мы это позже настроим). А nginx соответственно отвечать на запросы к этой картинке, забирая ее из папки media. В случае со статикой практически то же самое, за исключением одного нюанса - django будет наполнять папку collected_static с помощью команды manage.py collectstatic (настроим в следующем пункте)

12) Настраиваем статику. Создаем папку static внутри проекта (в папке src)

```
(env) vagrant@precise64:~/stackoverflow$ cd src/
(env) vagrant@precise64:~/stackoverflow/src$ ls
application db.sqlite3 manage.py requirements.txt
(env) vagrant@precise64:~/stackoverflow/src$ mkdir static
(env) vagrant@precise64:~/stackoverflow/src$ ls
application db.sqlite3 manage.py requirements.txt static
(env) vagrant@precise64:~/stackoverflow/src$
```

Именно в эту папку мы будем класть какие-либо картинки, файлы стилей и прочее, то, что нам нужно будет для отображения страниц проекта (я говорю не про верстку, то есть не .html-файлы, а именно файлы ресурсов (картинки)). Теперь настраиваем django-проект так, чтобы он искал статические файлы в папке static, и складывал их специально для nginx в папку collected_static. Для этого лезем редактировать настройки проекта (файлы application/settings.py). Дописываем в конец файла:

```
STATIC_URL = '/static/'
STATIC_ROOT = '/home/vagrant/stackoverflow/collected_static/'
STATICFILES_DIRS = ('/home/vagrant/stackoverflow/src/static/', )
```

Прописываете пути к своим папкам. Настройка STATICFILES_DIRS указывает django-проекту, где именно в исходниках он должен искать статику. Настройка STATIC_ROOT показывает, куда он должен скопировать всю эту статику по команде collectstatic. Такая хитрожопая схема копирования статики применяется в django потому, что некоторые python-библиотеки являются django-приложениями и притаскивают собственный набор картинок и прочего. Таким образом, когда мы делаем manage.py collectstatic - мы на самом деле копируем в нужную для nginx папку не только наши собственные файлы, но и все прочие картинки, использующиеся другими django-приложениями, включенными в наш проект.

13) Собственно, давайте попробуем collectstatic. Кладем в static пустой файлик для теста и запускаем.

```
(env) vagrant@precise64:~/stackoverflow/src$ touch static/blabla.gif
(env) vagrant@precise64:~/stackoverflow/src$ ls static/
blabla.gif
(env) vagrant@precise64:~/stackoverflow/src$ ./manage.py collectstatic

You have requested to collect static files at the destination
location as specified in your settings:

/home/vagrant/stackoverflow/collected_static

This will overwrite existing files!
Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes
Copying '/home/vagrant/stackoverflow/src/static/blabla.gif'
Copying '/home/vagrant/stackoverflow/env/local/lib/python2.7/site-
packages/django/contrib/admin/static/admin/img/calendar-icons.svg'
...
57 static files copied to '/home/vagrant/stackoverflow/collected_static'.
(env) vagrant@precise64:~/stackoverflow/src$ ls ~/stackoverflow/collected_static/
admin blabla.gif
```

Как видите, django скопировала не только наш файлик, но и еще 56 файлов с различной фигней, которая нужна для работы приложения django-admin (оно включено в проект по умолчанию). Если впоследствии добавим еще какое-нибудь стороннее django-приложение в проект - оно обязательно притащит еще какое-то количество своих статических файлов, которые мы точно также сможем одной командой скинуть туда, где их сможет раздавать nginx

14) С установкой и предварительной настройкой django мы наверное всё. Поставим утилитку tree и посмотрим, что у нас примерно должно было

получиться в смысле файловой структуры

```
(env) vagrant@precise64:~/stackoverflow$ sudo apt-get install tree
...
(env) vagrant@precise64:~/stackoverflow$ tree -L 2
.
|-- collected_static
| |-- admin
| `-- blabla.gif
|-- env
| |-- bin
| |-- include
| |-- lib
| |-- local
| `-- pip-selfcheck.json
|-- src
| |-- application
| |-- db.sqlite3
| |-- manage.py
| |-- requirements.txt
| `-- static
`-- uploads
```

Вот как-то так. Обращаю внимание - это не разработка, это только предварительная установка и настройка django так, чтобы впоследствии можно было сделать из этого нормальный проект.

Установка и настройка nginx

1) Собственно, всё просто и банально. Ставим с помощью apt-get. (Если apt-get не найдет каких-то пакетов, вам поможет apt-get update)

```
(env) vagrant@precise64:~/stackoverflow$ sudo apt-get install nginx
Reading package lists... Done
...
(env) vagrant@precise64:~/stackoverflow$
```

2) Всё, nginx установлен и запущен в данный момент. Пробуем его остановить, запустить, ребутнуть...

```
(env) vagrant@precise64:~/stackoverflow$ sudo /etc/init.d/nginx stop
Stopping nginx: nginx.
(env) vagrant@precise64:~/stackoverflow$ sudo /etc/init.d/nginx start
Starting nginx: nginx.
(env) vagrant@precise64:~/stackoverflow$ sudo /etc/init.d/nginx restart
Restarting nginx: nginx.
```

```
(env) vagrant@precise64:~/stackoverflow$ sudo /etc/init.d/nginx reload
Reloading nginx configuration: nginx.
(env) vagrant@precise64:~/stackoverflow$
```

3) Пробуем заглянуть на http://localhost/ и убедиться, что nginx работает

4) Основной конфиг nginx лежит в папке /etc/nginx/nginx.conf, но он нам особо не нужен - лучше мы создадим конфиг для своего проекта в /etc/nginx/conf.d/stackoverflow.conf (попутно убиваем дефолтный конфиг сайта, который идет вместе с сервером). После чего убеждаемся, то мы ничего не сломали и nginx будет работать

```
(env) vagrant@precise64:~/stackoverflow$ sudo touch
/etc/nginx/conf.d/stckoverflow.conf
(env) vagrant@precise64:~/stackoverflow$ sudo rm /etc/nginx/sites-available/default
(env) vagrant@precise64:~/stackoverflow$ sudo rm /etc/nginx/sites-enabled/default
(env) vagrant@precise64:~/stackoverflow$ sudo /etc/init.d/nginx configtest
Testing nginx configuration: nginx.
(env) vagrant@precise64:~/stackoverflow$
(env) vagrant@precise64:~/stackoverflow$ sudo /etc/init.d/nginx restart
Restarting nginx: nginx.
```

5) Правим конфиг, за основу берем примерно вот такой файлик https://github.com/sat2707/web/blob/master/tcp_servers/stackoverflow.conf (предполагается, что сам django-сервер, сам или через gunicorn будет запущен у вас на порту 8080, а nginx будет слушать на дефолтном 80м порту, и проксировать часть запросов на порт 8080 плюс раздавать статические файлы при запросе к урлам /static/ и /media/)

Собственно всё, это всё, что вы должны сделать к среде. Довольно объемно, но весьма просто, особенно учитывая, что у вас есть готовые рецепты. Удачи с ДЗ, если есть вопросы - задавайте в каментах, либо в личке.

Всем, кто до сих пор довольно поверхностно представляет себе синтаксис языка python - в обязательном порядке сюда http://ru.diveintopython.net/toc.html .

Разбалловка.

Максимум за ДЗ - 15 баллов
django - 10 баллов, nginx - 5 баллов

Если вдруг кто-нибудь напряжется и поставит-настроит в качестве СУБД mysql (вместо стандартно идущего вместе с django sqlite) - накину еще 5 баллов сверху :)