Instrucciones para implantar la plataforma de Prácticas

Versión 1.4 2018/10/26

Paquetes necesarios

El software instalado en los laboratorios es Ubuntu 18.04, con los siguientes paquetes relevantes:

- postgresql
- apache2
- python
- Flask
- SQLAlchemy
- virtualenv
- tora
- phppgadmin
- pgadmin3
- google-chrome-stable
- firefox
- xsltproc
- libxml2-utils

Los pasos necesarios para instalarlos:

- Como tu usuario de escritorio, ejecuta:

```
$ sudo apt-get install postgresql apache2 \
tora phppgadmin libqt4-sql-psql \
google-chrome-stable pgadmin3 firefox xsltproc libxml2-utils \
python python-pip \
libapache2-mod-wsgi python-flask jq python-virtualenv \
python-sqlalchemy python-flask-sqlalchemy python-sqlalchemy-utils \
python-psycopg2
```

  (si el paquete google-chrome da problemas, se puede descargar de la web de google)

- Para la instalacion de Python, VirtualEnv y PyCharm también se pueden consultar las instrucciones de los enlaces del curso:

- Instalación de algunas de las herramientas necesarias para las prácticas (Linux/MacOs)

Se recomienda el uso de las herramientas de desarrollador Web que tanto Firefox como Chrome tienen incluidas

Configurar PostgreSQL

- Para que podamos utilizar el usuario "alumnodb", es necesario editar el fichero /etc/postgresql/10/main/pg_hba.conf para que contenga las siguientes líneas adicionales. **Es importante que estas líneas estén situadas delante de otras directivas de autenticación (por ejemplo: al principio del fichero).**

  ```
  local       all     alumnodb                        md5
  host        all     alumnodb    127.0.0.1/32        md5
  ```

  También se puede usar 'trust' en lugar de 'md5' como método alternativo de autenticación (no se pide contraseña).

- Asimismo, conviene editar el fichero /etc/postgresql/10/main/postgresql.conf, buscar las líneas con los valores comentados de autovacuum_vacuum_threshold, autovacuum_analyze_threshold y establecer los valores:

  ```
  autovacuum_vacuum_threshold = 5000000
  autovacuum_analyze_threshold= 5000000
  ```

- Tras esto, es necesario reiniciar el servicio ejecutando:

  ```
  $ sudo systemctl restart postgresql
  ```

- Ahora debemos crear el "rol" alumnodb en postgres. Para ello, iniciamos sesión como usuario "postgres" y creamos el rol con los siguientes comandos:

  ```
  $ sudo su - postgres
  ```

- y después:

  ```
  $ createuser -s alumnodb
  ```

- Para dar de alta el lenguaje plpgsql en todas las nuevas bases de datos, como usuario postgres, ejecutar (normalmente no es necesario porque ya está dado de alta):

  ```
  $ createlang plpgsql template1
  ```

- Para acceder a la base de datos, y asignar una contraseña (necesaria para acceder desde phppgadmin; introducir 'alumnodb' como contraseña)

```
$ psql
\password alumnodb
...
\q
```

- Por último, para comprobar que todo está correcto, creamos una base de datos. Como nuestro usuario de inicio de sesión, ejecutamos:

  ```
  $ createdb -U alumnodb bdat
  ```

También conviene instalar el programa schemaSpy, http://schemaspy.sourceforge.net, (y schemaSpyGUI, http://sourceforge.net/projects/schemaspygui) para obtener diagramas de las bases de datos desarrolladas.

Creación de un entorno python con Flask

Para la creación en un entorno python en el que probar páginas desarrolladas en Flask, ejecutar los siguientes mandatos:

```
$ virtualenv -p python2 si1pyenv

$ source si1pyenv/bin/activate

$ pip install Flask SQLAlchemy Flask-SQLAlchemy SQLAlchemy-Utils \
  psycopg2 'itsdangerous<1.0.0' Flask-Session

$ deactivate
```

Nota importante: Las prácticas se realizarán con python versión 2. En los laboratorios, la versión de python por defecto es la 3, por ello es preciso indicar la opción **`-p python2`** al crear el entorno de python con virtualenv.

Nota importante: El paquete Flask-Session no es compatible con la versión más reciente de itsdangerous, por ello es preciso indicar **`'itsdangerous<1.0.0'`** para instalar una versión anterior compatible.

Para probar el entorno, crear el fichero *hello.py* con el siguiente contenido:

```
from flask import Flask
```

```
app = Flask(__name__)
@app.route("/")

def hello():
    return "Hello World!"
if __name__ == "__main__"
:
    app.run(host='0.0.0.0', port=5001, debug=True)
```

A continuación, ejecutar el programa con los siguientes mandatos:

```
$ source si1pyenv/bin/activate
$ python hello.py
```

Accediendo a http://localhost:5001 debería aparecer el mensaje 'Hello World!'

Nota importante: Téngase presente que al crear el entorno si1pyenv se introducen rutas absolutas en ficheros que se generan, principalmente en el directorio si1pyenv/bin. Esto hace que si se mueve (o se hace backup y se restaura) el directorio si1pyenv a otra localización dejará de funcionar.

Configurar Apache para que muestre páginas de usuario

Ahora vamos a configurar el servidor web Apache para publicar la carpeta personal public_html en la ruta http://localhost/~usuario/. Para ello:

- Crear el directorio "public_html" dentro de $HOME (carpeta personal: /home/*usuario*)

  ```
  $ mkdir ~/public_html
  ```

- Ejecutar este mandato:

  ```
  $ sudo a2enmod userdir
  ```

- Tras esto, es necesario reiniciar el servicio ejecutando:

  ```
  $ sudo service apache2 restart
  ```

- Ya se puede acceder al contenido de ésta en http://localhost/~usuario/, donde "usuario" es el que ha iniciado la sesión.

Configurar Flask bajo Apache

Para poder ejecutar páginas python es necesario instalar y configurar el módulo mod_wsgi de Apache según lo indicado más arriba (mandato '`sudo apt-get install …`' más arriba). También se puede consultar http://flask.pocoo.org/docs/0.12/deploying/mod_wsgi/

Para configurar el acceso a páginas python seguir los siguientes pasos:

- Ejecutar el siguiente mandato:

  ```
  $ sudo a2enmod wsgi
  ```

- Editar el archivo `/etc/apache2/mods-enabled/userdir.conf` y añadir ExecCGI a las opciones y el handler de wsgi:

  ```
  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec ExecCGI
  AddHandler wsgi-script .wsgi
  ```

- Tras esto, es necesario reiniciar el servicio ejecutando:

  ```
  $ sudo systemctl restart apache2
  ```

Publicar un aplicación Flask con su entorno virtual propio bajo Apache

Como hemos indicado anteriormente, para publicar páginas web desarrolladas en python es necesario el módulo de Apache mod_wsgi, que hace de puente entre el servidor Web (Apache) y la aplicación web en python (Flask).

Para ello es necesario crear un fichero .wsgi como el siguiente, al que llamaremos *hello.wsgi*, que además hace uso del entorno virtual creado anteriormente:

```
#!/usr/bin/python

import os
import sys

# virtualenv
this_dir = os.path.dirname(os.path.abspath(__file__))
activate_this = this_dir + '/si1pyenv/bin/activate_this.py'
execfile(activate_this, dict(__file__=activate_this))
```

```
# anhadir dir de este fichero a path
sys.path.insert(0, this_dir)

from hello import app as application
```

Si colocamos el fichero hello.py y este nuevo hello.wsgi en el directorio public_html y creamos aquí un entorno si1pyenv según las instrucciones que ya hemos visto, accediendo a la URL http://localhost/~<usuario>/hello.wsgi nos aparecerá el mensaje 'Hello World!' de la aplicación.

Trucos de PostgreSQL

Se recomienda el uso de las herramientas de gestión:

- $ tora
- $ pgadmin3
- http://localhost/phppgadmin/

Para entrar en el entorno interactivo de psql y obtener ayuda:

```
$ psql –U alumnodb bdat
bdat=#help
bdat=#\help
bdat=#\?
bdat=#\q
```

Para evitar tener que introducir usuario, contraseña y base de datos:

```
$ export PGUSER=alumnodb

$ export PGPASSWORD=alumnodb
$ export PGDATABASE=si1
```

Para ejecutar un script:

```
$ cat miscript.sql | psql –U alumnodb bdat
```

Para crear una base de datos:

```
$ createdb –U alumnodb dbname
```

Para volcar el contenido (y estructura) de una base de datos:

```
$ pg_dump –U alumnodb bdat > outputfile.sql
```

El fichero resultante es un script que se puede usar para recrear la base de datos, ejecutándolo tal y como se indica más arriba.

Para realizar una carga masiva de datos se puede usar el mandato COPY. Ver el contenido del script de salida del mandato pg_dump.

```
create table profesor (id int, nombre char(20), apellido char(20));
COPY profesor from stdin using delimiters '|';
1|Pedro|Pascual
2|Julia|Díaz
\.
```

Para ver los logs de postgresql:

```
$ sudo su – postgres
$ ls -al /var/log/postgresql/
$ cat /var/log/postgresql/postgresql*.log
```

Enlaces de referencia

- http://www.w3schools.com
- http://www.python.org
- http://www.postgresql.org
- http://flask.pocoo.org
- https://www.sqlalchemy.org/