Virtualenvwrapper, LiClipse, pip y algunas cosas mas

2 de Diciembre de 2016 a las 23:04

Todo programador debe tener un entorno de trabajo que resulte cómodo y que facilite el desarrollo de aplicaciones, aquí incluimos diversas herramientas y técnicas. Como todo no es picar código, estas herramientas facilitan el desarrollo, consiguiendo un desarrollo mas eficiente.

En este articulo, veremos como crear un entorno de trabajo para *Django* con *virtualenvwrapper(4.5.0)* y pip(7.1.2) e integrarlo con *Liclipse* (2.4.0), en un sistema *Fedora* 22.

VIRTUALENVWRAPPER

Esta herramienta es un conjunto de extensiones de otra herramienta llamada *virtualenv* que permite crear un entorno virtual de desarrollo para *Python*. *Virtualenvwrapper* crea un directorio, *virtualenvs*, donde almacenara los entornos virtuales que se creen, también almacena un conjunto de ejecutables para realizar diversas tareas.

Para crear un entorno virtual con *virtualwrapper* se utiliza el comando *mkvirtualenv*.

```
mkvirtualenv biblioteca
```

Esto creara un entorno virtual denominado biblioteca. En mi caso, como utilizo *Python* 3, utilizo la opción *-p* que permite especificar un interprete.

```
mkvirtualenv biblioteca -p /usr/bin/python3.4
```

El entorno virtual creado utilizara el interprete de *Python* versión 3.4.

Si nos situamos en el directorio *.virtualenvs* veremos un directorio *biblioteca* donde se ha creado un entorno compuesto por una estructura de directorios donde almacenara el interprete, librerías, paquetes y otras herramientas. En este directorio sera donde debemos crear la estructura para nuestra aplicación.

Cada entorno virtual creado, tendrá su propio directorio y una estructura de directorios, que serán independiente unos de otros. Podremos tener proyectos con diferentes versiones de *Python*, librerías y paquetes, evitando tener que instalar todo en el sistema.

La primera vez que ejecutamos *mkvirtualenv*, crea el entorno y lo activa, en el *prompt* de la *shell* aparecerá el nombre del entorno entre paréntesis. Para salir del entorno ejecutamos el comando.

```
deactivate
```

Para activar un entorno en cualquier momento, debemos especificar su nombre.

```
workon biblioteca
```

Para mostrar todos los entornos creados.

```
lsvirtualenv
```

Podemos personalizar el comportamiento de *virtualenwrapper*, modifcando una serie de ficheros que permiten cambiar el comportamiento de sus comandos*, e*stos ficheros se encuentran en el directorio *.virtualenvs.*

En mi caso, modifico el fichero *postmkvirtualenv* que permite modificar el comportamiento posterior a ejecutar *mkvirtualenv*. Añado las siguientes lineas.

```
#!/bin/bash
# This hook is sourced after a new virtualenv is activated.
proj_name=$(basename $VIRTUAL_ENV)
mkdir $HOME/.virtualenvs/proyectos/$proj_name
add2virtualenv $HOME/.virtualenvs/proyectos/$proj_name
cd $HOME/.virtualenvs/proyectos/$proj_name
```

Estas lineas, crearan un directorio con el nombre especificado con el comando *mkvirtualenv* dentro de *.virtualenvs/proyectos*. Cada vez que creamos un entorno virtual , lo creara en la ruta especificada y nos situara en el directorio del entorno creado.

Esto último, solo afecta en la creación del entorno, pero no afectara cuando solo activemos el entorno, comando *workon*, que tiene asociado otro fichero para modificar su comportamiento. Este fichero es *postactivate* , que se encuentra dentro del directorio *virtualenvs*.

Estas lineas permiten situarnos en el directorio del proyecto especificado después de la activación.

Existen mas ficheros que permiten personalizar el comportamiento de los comandos*,* todos están dentro del directorio *virtualenvs,* consultar la documentación de *virtualenwapper* para obtener mas información.

PIP

Después de configurar un entorno para nuestro proyecto, lo primero que tendremos que hacer es instalar *Django*,para ello utilizaremos un gestor de paquetes que viene incluido entre los comandos del entorno virtual, denominada *Pip.*

Esta herramienta permite instalar paquetes de una forma muy fácil y también realizar otras tareas de administración. Estos paquetes se encuentran en un repositorio de *pypi* , donde son almacenados una gran cantidad de paquetes para python que podremos utilizar en nuestros proyectos.

Instalar un paquete con *pip* es muy fácil solo hay que utilizar la opción *install*. Por ejemplo para instalar *Django.*

```
pip install Django
```

Instalando *Django* en el sistema, si tenemos activado un entorno virtual, solo se instalará en este entorno y fuera del entorno virtual no tendrá efecto. El comando *pip* tiene muchas opciones, no es motivo de este artículo explicarlas aquí, para mas información ir a la documentación en el siguiente enlace (https://pip.pypa.io/en/latest/).

En el desarrollo de una proyecto de *Django*, lo mas seguro, será necesario utilizar una serie de librerías u otras utilidades que no viene por defecto, que deberán ser instaladas con el comando *pip*. Si utilizamos los entornos virtuales, podemos obtenerr una lista con todas las librerías y utilidades utilizadas para ese proyecto.

Si un proyecto debe migrarse a otro lugar podemos llevar esa lista para saber los paquetes que debemos instalar. Para facilitar esa tarea podemos crear un fichero *requirements.txt*, que puede utilizar cualquier nombre pero el anterior es el más utilizado.

Este fichero se puede crear con el comando *pip* y la opción *freeze.*

```
pip freeze > requirements.txt
```

Este comando descargara e instalara todos los paquetes incluidos en el fichero *requirements* en un nuevo entorno, que debe ser activado previamente para una migración correcta.

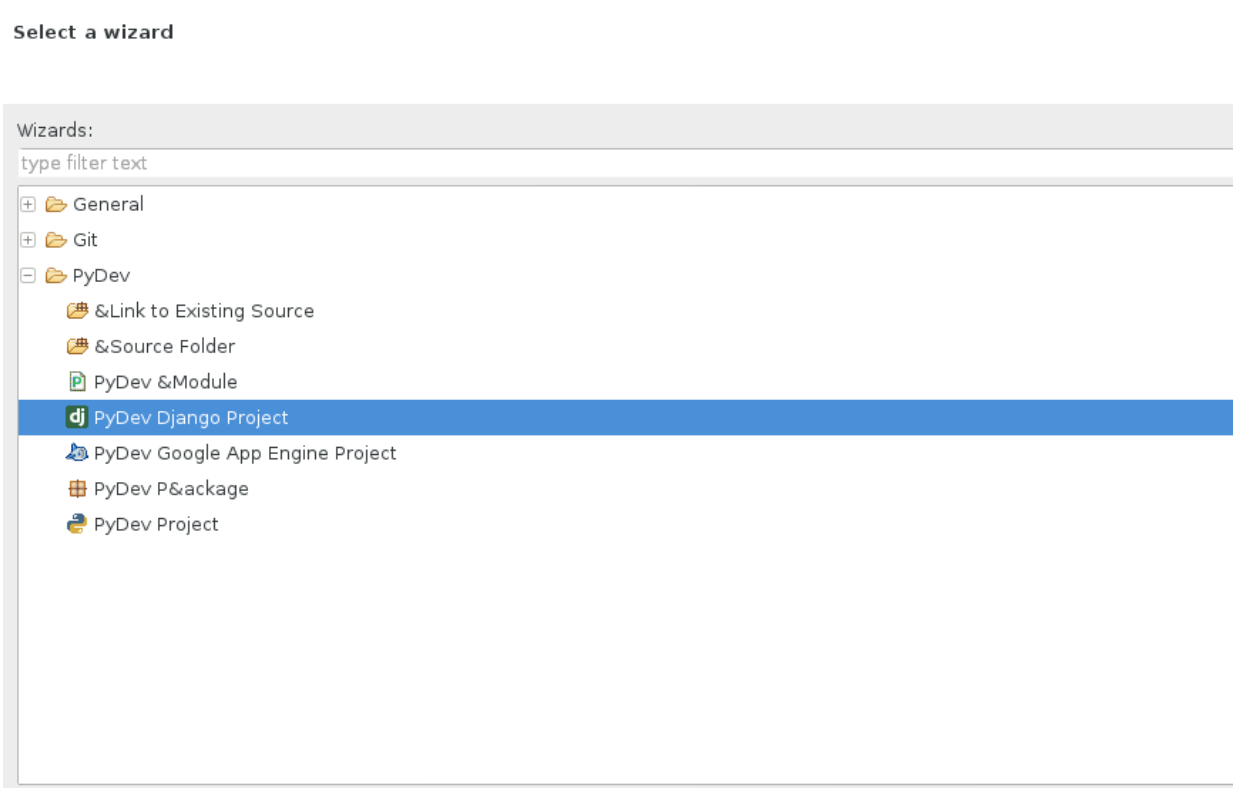
LICLIPSE

Ahora tenemos un entorno virtual creado con los paquetes que necesitamos instalados con *pip.* Lo siguiente sera empezar a programar con la ayuda de algún editor de código o IDE.

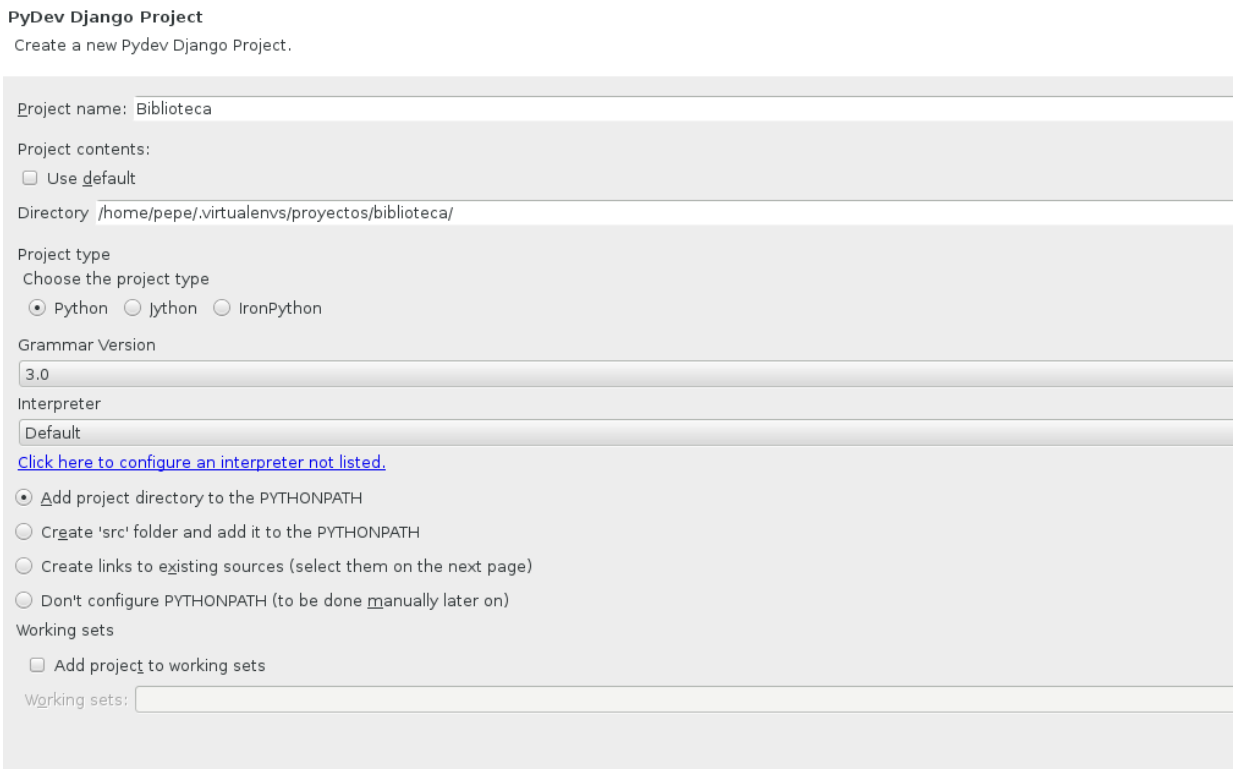
Yo utilizo para programar Liclipse , que es un IDE basado en Eclipse con el plugin PyDev instalado, que proporciona un entorno de desarrollo para Python.

Lo siguiente, sera integrar el entorno virtual creado anteriormente con *Liclipse,* como se había creado un entorno denominado *Biblioteca* sera utilizado para crear un proyecto en *Django* en *Liclipse*.

Primero será la instalación de *Liclipse* que es muy fácil, en la página oficial hay instrucciones detalladas. Una vez instalado y ejecutado, creamos un proyecto nuevo (Django) siguiendo la siguiente ruta *File->New->Other*, buscamos *PyDev->PyDev Django Project.*

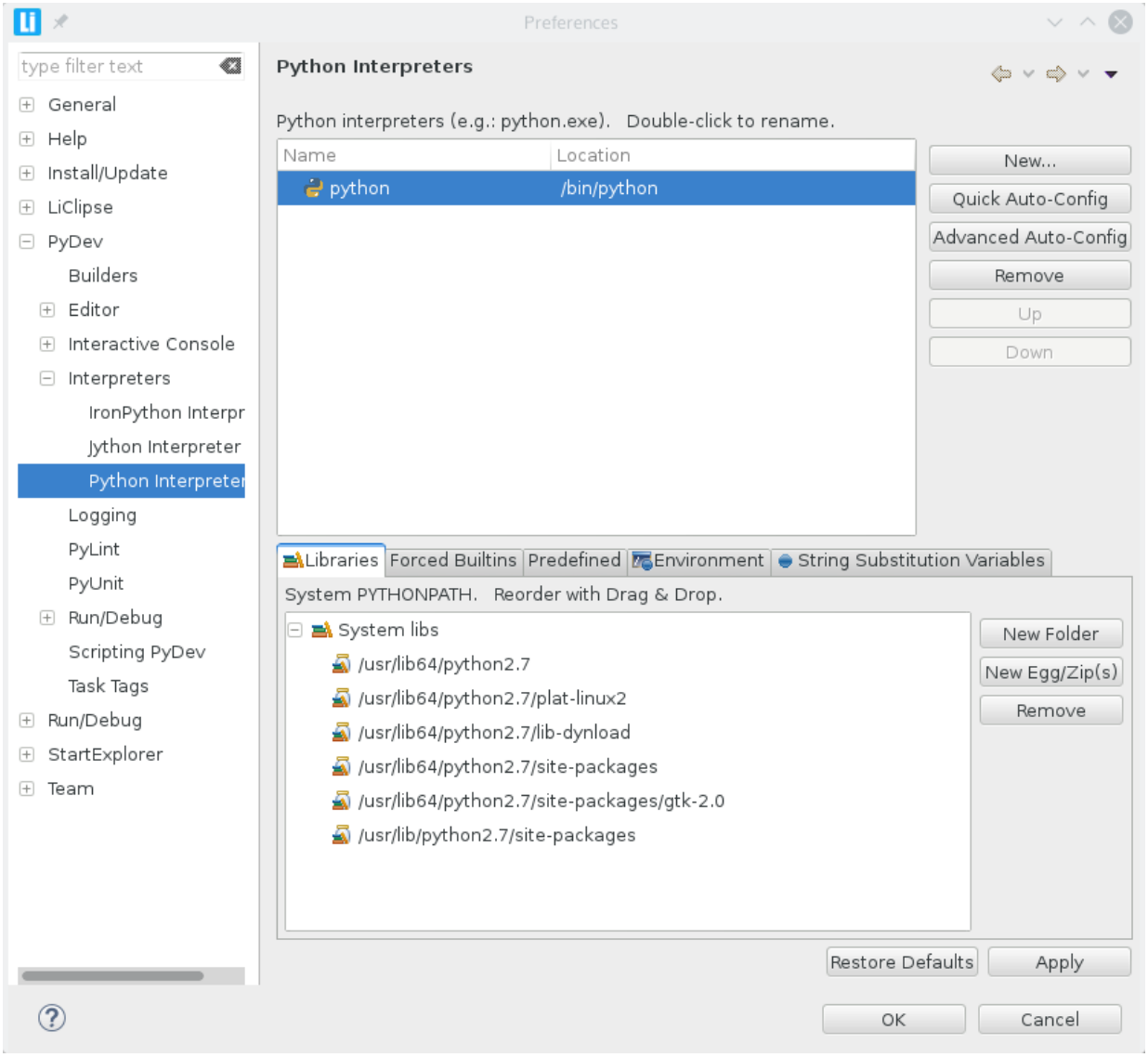


En la siguiente ventana, escoger en nombre del proyecto ,*Project name,* la ruta donde se almacenara el código del proyecto. Si la ruta por defecto no es correcta, se debe desmarcar esta opción y escribir la ruta correcta.

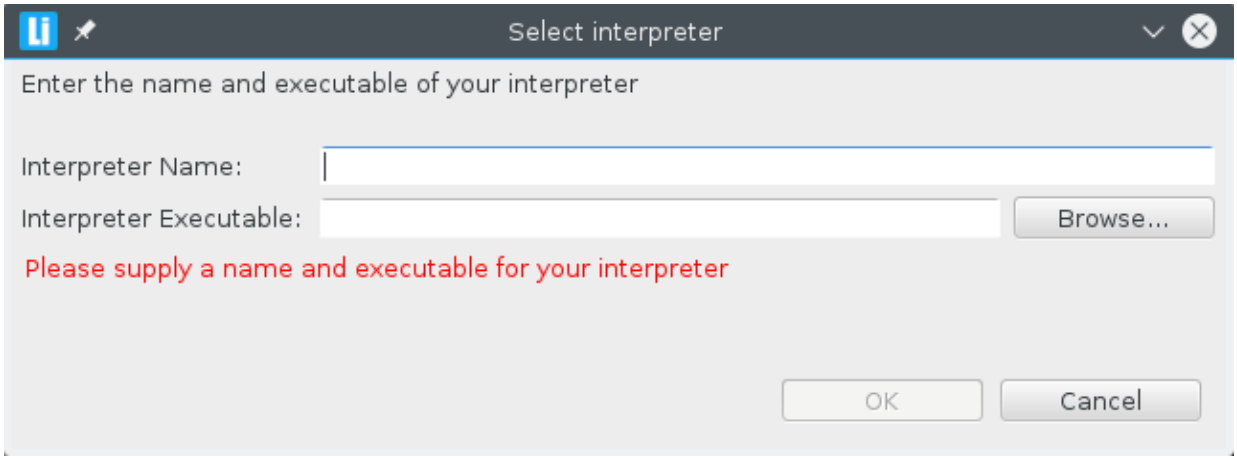


En la parte del interprete de python, *Grammar Version* y *Interprete*, debemos escoger

el que incluye el entorno virtual, primero escogemos la versión de python que utilizaremos y para el interprete pulsamos el enlace *Click here to configure an interpreter not listed,*para añadir nuestro interprete a la lista.



En la siguiente ventana aparecen todos los interpretes que tiene configurado *Liclipse*,debemos añadir el intérprete que tenemos en nuestro entorno virtual. Para añadir uno nuevo , pulsamos en el boton *New.*



Se deben especificar un nombre de interprete, *Interpreter Name*, para identificarlo y la ruta donde se encuentra el interprete, que se encuentra en el directorio *bin* dentro del entorno virtual. Escogido esos dos elementos, pulsamos *Ok*.

En la siguiente ventana, muestra las carpetas que podemos incluir, seleccionar todas para que el interprete pueda utilizar los paquetes que instalemos en el entorno mediante *e l* comando *p i p* y aceptamos. Volvemos a la ventana anterior, donde pulsaremos *Apply* para confirmar los cambios . Ahora en la lista de interpretes podemos seleccionar el interprete definido anteriormente.

En las siguientes ventanas , podemos escoger si nuestro proyecto lo queremos incluir en algún proyecto que ya este creado y la configuración de la base de datos del proyecto, por defecto aparece la configuración para SQLite.

Con esto, ya tenemos un entorno de desarrollo para Django, un directorio, que es independiente de otros entornos que tengamos creados y un conjunto de herramientas que permitirán administrarlo de forma fácil.

Este entorno es algo simple, podemos mejorarlo creado varios entornos virtuales como; *desarrollo*, *testing* y *producción*, que tendrán sus propios requisitos. Para saber mas como realizar esto, recomiendo el siguiente tutorial donde lo explican muy bien, taskbuster-django-tutorial.