**Inicio rápido: crear la primera aplicación web de Python con Visual Studio**

* 26/06/2018
* Tiempo de lectura: 14 minutos
* Colaboradores
  + [Kraig Brockschmidt](https://github.com/kraigb)

* + [](https://github.com/olprod)

En esta introducción a Visual Studio como un IDE de Python, con una duración de entre 5 y 10 minutos, creará una sencilla aplicación web Python basada en el marco Flask. El proyecto se crea mediante discretos pasos que le ayudarán a conocer las características básicas de Visual Studio.

Si todavía no ha instalado Visual Studio, vaya a la página [Descargas de Visual Studio](https://visualstudio.microsoft.com/downloads/?utm_medium=microsoft&utm_source=docs.microsoft.com&utm_campaign=button+cta&utm_content=download+vs2017) para instalarlo de forma gratuita. En el instalador, asegúrese de seleccionar la carga de trabajo **Desarrollo de Python**.

**Crear el proyecto**

Los pasos siguientes crean un proyecto vacío que actúa como un contenedor para la aplicación:

1. Abra Visual Studio 2017.
2. En la barra de menús superior, elija **Archivo > Nuevo > Proyecto**.
3. En el diálogo cuadro **Nuevo proyecto**, escriba "Proyecto web de Python" en el campo de búsqueda situado en la parte superior derecha, elija **Proyecto Web** en la lista del medio, asigne un nombre al proyecto (por ejemplo, "HelloPython") y luego elija **Aceptar**.

Si no ve las plantillas de proyecto de Python, cancele haciendo clic fuera del cuadro de diálogo **Nuevo proyecto** y, en la barra de menús superior, elija **Herramientas > Obtener herramientas y características** para abrir el **instalador de Visual Studio**. Elija la carga de trabajo **Desarrollo de Python** y, luego, haga clic en **Modificar**.

1. El nuevo proyecto se abre en el panel de la derecha del **Explorador de soluciones**. En este momento el proyecto está vacío porque no contiene ningún otro archivo.

**Pregunta: ¿Cuál es la ventaja de crear un proyecto en Visual Studio para una aplicación de Python?**

**Respuesta**: Las aplicaciones de Python suelen definirse con el uso de carpetas y archivos únicamente, pero esta estructura sencilla puede ser más pesada a medida que las aplicaciones aumentan de tamaño y se generan posibles archivos de forma automática, JavaScript para aplicaciones web, etc. Un proyecto de Visual Studio le ayuda a administrar esta complejidad. El proyecto (un archivo *.pyproj*) identifica todos los archivos de origen y de contenido asociados al proyecto, contiene información de compilación para cada archivo, mantiene la información para integrarse con sistemas de control de código fuente y le ayuda a organizar la aplicación en componentes lógicos.

**Pregunta: ¿Qué es la "solución" que se muestra en el Explorador de soluciones?**

**Respuesta**: Una solución de Visual Studio es un contenedor que le ayuda a administrar uno o más proyectos relacionados como un grupo y almacena los valores de configuración que no son específicos de un proyecto. Los proyectos de una solución también pueden hacerse referencia entre sí, de modo que la ejecución de un proyecto (una aplicación de Python) crea automáticamente un segundo proyecto (por ejemplo, una extensión de C++ usada en la aplicación de Python).

**Instalación de la biblioteca de Flask**

En las aplicaciones web de Python casi siempre se usa una de las muchas bibliotecas de Python disponibles para controlar detalles de bajo nivel como el enrutamiento de solicitudes web y la forma de las respuestas. Para ello, Visual Studio proporciona una variedad de plantillas para aplicaciones web, una de las cuales se usará más adelante en este inicio rápido.

Aquí, utilice los pasos siguientes para instalar la biblioteca de Flask en el "entorno global" predeterminado que usa Visual Studio para este proyecto.

1. Expanda el nodo **Entornos de Python** en el proyecto para ver el entorno predeterminado para el proyecto.
2. Haga clic con el botón derecho en el entorno y seleccione **Instalar paquete de Python**. Este comando abre la ventana **Entornos de Python** por la pestaña **Paquetes**.
3. Escriba "flask" en el campo de búsqueda y seleccione **instalación de PIP flask desde PyPI**.Acepte los mensajes de privilegios de administrador y observe el progreso en la ventana **Salida**de Visual Studio. (Se le pedirá confirmación de elevación cuando la carpeta de paquetes del entorno global esté ubicada en un área protegida, como *C:\Archivos de programa*).
4. Una vez instalada, la biblioteca aparece en el entorno en el **Explorador de soluciones**, lo que significa que se puede usar en código de Python.

**Nota**

En lugar de instalar las bibliotecas en el entorno global, los desarrolladores suelen crear un "entorno virtual" en el que se instalan las bibliotecas de un proyecto específico. Las plantillas de Visual Studio normalmente ofrecen esta opción, como se describe en [**Inicio rápido: Crear un proyecto de Python desde una plantilla en Visual Studio**](https://docs.microsoft.com/es-es/visualstudio/python/quickstart-02-python-in-visual-studio-project-from-template).

**Pregunta: ¿Dónde obtener más información acerca de los otros paquetes de Python disponibles?**

**Respuesta**: Visite el [índice de paquete de Python](https://pypi.org/).

**Agregar un archivo de código**

Ahora está listo para agregar un poco de código de Python para implementar una aplicación web básica.

1. Haga clic con el botón derecho en el **Explorador de soluciones** y seleccione **Agregar > Nuevo elemento**.
2. En el cuadro de diálogo que aparece, seleccione **Archivo de Python vacío**, asígnele el nombre *app.py* y seleccione **Agregar**. Visual Studio abre automáticamente el archivo en una ventana del editor.
3. Copie el siguiente código y péguelo en *app.py*:

PythonCopiar

from flask import Flask

# Create an instance of the Flask class that is the WSGI application.

# The first argument is the name of the application module or package,

# typically \_\_name\_\_ when using a single module.

app = Flask(\_\_name\_\_)

# Flask route decorators map / and /hello to the hello function.

# To add other resources, create functions that generate the page contents

# and add decorators to define the appropriate resource locators for them.

@app.route('/')

@app.route('/hello')

def hello():

# Render the page

return "Hello Python!"

if \_\_name\_\_ == '\_\_main\_\_':

# Run the app server on localhost:4449

app.run('localhost', 4449)

1. Puede haber observado que el cuadro de diálogo **Agregar > Nuevo elemento** muestra muchos otros tipos de archivos que puede agregar a un proyecto de Python, como una clase de Python, un paquete de Python, una prueba unitaria de Python, archivos *web.config*, etc. En general, estas plantillas de elemento, tal y como se denominan, son una excelente manera de crear rápidamente archivos con código reutilizable útil.

**Pregunta: ¿Dónde puedo obtener más información sobre Flask?**

**Respuesta**: Consulte la documentación de Flask, empezando por el [inicio rápido de Flask](http://flask.pocoo.org/docs/0.12/quickstart/#quickstart).

**Ejecutar la aplicación**

1. Haga clic con el botón derecho en *app.py* en el **Explorador de soluciones** y seleccione **Establecer como archivo de inicio**. Este comando identifica el archivo de código que se va a iniciar en Python cuando se ejecuta la aplicación.
2. Haga clic con el botón derecho en el proyecto en el **Explorador de soluciones** y seleccione **Propiedades**. Después, haga clic en la pestaña **Depurar** y establezca la propiedad **Número de puerto** en 4449. Este paso garantiza que Visual Studio inicia un explorador con localhost:4449 para que coincida con los argumentos app.run en el código.
3. Seleccione **Depurar > Iniciar sin depurar** (**Ctrl**+**F5**) para guardar los cambios en los archivos y ejecutar la aplicación.
4. Aparece una ventana de comandos con el mensaje "\* Ejecutando en <https://localhost:4449/>" y se abre una ventana del explorador para localhost:4449 en la que se puede ver el mensaje "Hello, Python!". La solicitud GET también aparece en la ventana de comandos con un estado de 200.

Si un explorador no se abre automáticamente, inicie el explorador que quiera y vaya a localhost:4449.

Si solo ve el shell interactivo de Python en la ventana Comandos, o bien si esa ventana parpadea brevemente en la pantalla, asegúrese de haber establecido *app.py* como archivo de inicio anteriormente en el paso 1.

1. Vaya a localhost:4449/hello para comprobar que el elemento decorador para el recurso /hello también funciona. Una vez más, la solicitud GET aparece en la ventana de comandos con un estado de 200. No dude en probar algunas otras direcciones URL para ver que muestran los códigos de estado 404 en la ventana de comandos.
2. Cierre la ventana Comandos para detener la aplicación. Después, cierre la ventana del explorador.

**Pregunta: ¿Cuál es la diferencia entre el comando Iniciar sin depurar e Iniciar depuración?**

**Respuesta**: **Iniciar depuración** se usa para ejecutar la aplicación en el contexto del [depurador de Visual Studio](https://docs.microsoft.com/es-es/visualstudio/python/debugging-python-in-visual-studio), lo que permite establecer puntos de interrupción, examinar las variables y recorrer el código línea por línea. Las aplicaciones se pueden ejecutar más lentamente en el depurador debido a los distintos enlaces que hacen posible la depuración. **Iniciar sin depurar**, en cambio, ejecuta la aplicación directamente como si la ejecutara desde la línea de comandos, sin contexto de depuración, y también inicia automáticamente un explorador y va a la dirección URL especificada en la pestaña **Depurar** de las propiedades del proyecto.

**Pasos siguientes**