Azure Functions in Python

# Introduzione

Alla vasta e rapida diffusione di modelli di programmazione serverless per applicazioni event-driven Microsoft ha risposto con le Azure Functions. Questo tipo di prodotti è conosciuto come Faas (function as a service), fornendo direttamente un ambiente dove memorizzare ed eseguire il proprio applicativo.

Fin da subito, l’integrazione con Python è stata una delle funzionalità più richieste. Tale connubio ha aumentato le possibilità di integrazione del servizio offrendo un ampio ventaglio di soluzioni basate su machine learning, data manipulation, scripting, automazione e molto altro.

In questo articolo troveremo un esempio di Azure Function basata su Python che mostrerà il potenziale in rapporto alla semplicità di sviluppo ed integrazione. In particolare, sarà mostrato come da un trigger Http vi è la possibilità di importare moduli esterni contenenti l’integrazione di librerie e pacchetti.

# Requisiti

* Account Azure
* Node.js
* Python 3.6 / 3.7 64 bit
* Visual Studio Code
* Python Extension per VSCode
* Azure Functions Extension per VSCode

# Struttura file progetto Azure Function

Le Azure Functions seguono un modello standard per l’impostazione del progetto che indica cosa verrà pubblicato su Azure.

\_\_app\_\_

| - function1

| | - \_\_init\_\_.py

| | - function.json

| | - module.py

| - function2

| | - \_\_init\_\_.py

| | - function.json

| - shared

| | - helper\_function1.py

| | - helper\_function2.py

| - host.json

| - requirements.txt

| - Dockerfile

Tests

Dove “\_\_app\_\_” è la cartella principale del progetto che al suo interno conterrà:

* *function1*: una cartella contenente i file necessari per la registrazione e l’esecuzione della funzione Python nominata “function1”.
* *Function2*: una cartella contenente i file necessari per la registrazione e l’esecuzione della funzione Python nominata “function2”.
* *shared*: una cartella contenente script Python di utility disponibili a tutte le funzioni che li importano.
* *hosts*.json: contiene i valori comuni ad ogni funzione dell’applicazione come, ad esempio, la versione di runtime delle Azure Functions. Alcune opzioni potrebbero non essere supportate nell’esecuzione locale.
* *requirements*.txt: contiene la lista di pacchetti che il sistema deve installare al momento della pubblicazione su Azure. Di default, sarà presente il pacchetto “azure-functions”.
* *Dockerfile*: utilizzato se l’applicazione dovesse essere pubblicata su un container specifico.

Sarà presente il file “local.settings.json”, utilizzato per contenere gli app settings (connection strings, etc…) solamente per l’esecuzione locale. Questo file non sarà pubblicato su Azure.

Inoltre, potranno essere presenti i file “.gitignore” per i file esclusi dal repository git e “.funcignore” per dichiarare i file che non dovranno essere pubblicati su Azure.

# Creazione tramite CLI

Tramite Node.Js, installare gli Azure Function Core Tools tramite il comando:

npm install -g azure-functions-core-tools

# Creazione tramite Azure Functions Extension

# Scrittura della Function

# Deploy su Azure, Monitoraggio ed Utilizzo

# Testing