



07_Python_7_Flask_Relazione

Citterio Giorgio e Colombo Umberto

Lo scopo di questa attività è rendere disponibili su Web i dati rilevati dal sensore, grazie all'utilizzo di Flask.

parte 1

Nella prima parte dell'attività dopo aver visto delle spiegazioni su come funziona HTML, abbiamo scritto un codice HTML per visualizzare una tabella con Data e ora e valori inseriti manualmente.

codice HTML tabella:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>parte1</title>
</head>
<style>
  table, th, td{
    border: 1px solid black;
  }
</style>
<body>
  <div id="container">
    <table style="width:75%">
      <tr>
        <th>Data e ora</th>
        <th>Valore</th>
      </tr>
      <tr>
        <td>2021-11-17 17.53.10</td>
        <td>125</td>
      </tr>
      <tr>
        <td>2021-11-17 17.54.10</td>
        <td>137</td>
      </tr>
      <tr>
        <td>2021-11-17 17.55.10</td>
        <td>175</td>
      </tr>
    </table>
  </div>
</body>
</html>
```

parte 2

L'obiettivo della seconda parte dell'attività è quello di prelevare i dati del sensore dal file JSON e scrivere un file HTML da Python per visualizzare questi dati sotto forma di tabella.

codice Python pyTohtml:

```
import json

with open('05_Python-5-JSON/parte3/datiSensore.json', 'r') as fp:
    lista = json.load(fp)
date = []
valoriSensori = []
stringa = ""
```

```

for i in range(len(lista)):
    date.append(lista[i]["DataOra"])
    valoriSensori.append(lista[i]["Valore"])
    stringa += "<tr><td>" + str(date[i]) + "</td><td>" + str(valoriSensori[i]) + "</td></tr>"
stringaFinale= "<style>table, th, td{border: 1px solid black;}</style><body><table><tr><th>Data e ora</th><th>Valore</th></tr>" + stringa + "</>
print (stringaFinale)
with open("07_Python-7-Flask/parte2/index.html", "w") as ft:
    ft.write(stringaFinale)

```

parte 3

Nella terza parte dell'attività abbiamo studiato il funzionamento del protocollo HTTP e successivamente abbiamo creato la prima applicazione Flask in grado di restituire la stringa HTML per generare la tabella.

codice Python flask1:

```

from flask import Flask
import json
import os

path = os.getcwd() + '/datiSensore.json'
app = Flask(__name__)
@app.route('/')

def returnHtml():
    with open(path, 'r') as fp:
        lista = json.load(fp)
        date = []
        valoriSensori = []
        stringa = ""
        for i in range(len(lista)):
            date.append(lista[i]["DataOra"])
            valoriSensori.append(lista[i]["Valore"])
            stringa += "<tr><td>" + str(date[i]) + "</td><td>" + str(valoriSensori[i]) + "</td></tr>"
        stringaFinale= "<!DOCTYPE html><html lang=en><head><meta charset=UTF-8><meta http-equiv=X-UA-Compatible content=IE=edge><meta name=
        print(stringaFinale)
        return stringaFinale

```

parte 4

In questa parte dell'attività abbiamo studiato come fare tunneling utilizzando i seguenti comandi.

```

ssh -N -4 -v -R 8106:172.17.3.95:5000 greppi@tunnel.vincenzov.net
tunnel.vincenzov.net:8106

```

parte 5

Nella quinta parte dell'attività abbiamo realizzato un test d'insieme di quanto fatto finora verificando il funzionamento del sistema in tutte le sue parti:

1. Programma sensore attivo su Arduino
2. Programma Python che riceve i dati del sensore e scrive, in continuazione, il file JSON
3. Applicazione Flask che legge il file JSON e produce il codice HTML
4. Tunnel per esporre il Web Server su Internet

parte 6 (approfondimenti)

L'ultima fase, quella di approfondimento, è divisa in 3 parti:

Nella prima bisognava imparare a migliorare la produzione di codice HTML utilizzando i template.

codice Python template1:

```

from flask import render_template
from flask import Flask
import json
import os

pathJ = os.getcwd()+'/datiSensore.json'
pathH = os.getcwd()+'/templates/index.html'
app = Flask(__name__)
@app.route('/')

def returnHtml():
    with open(pathJ, 'r') as fp:
        lista = json.load(fp)
    return render_template('index.html', dizValori=lista)
if __name__=="__main__":
    app.run(debug=True)

```

template HTML:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="refresh" content="1">
  <title>Dati sensore</title>
  <style>table, th, td{border: 1px solid black;}</style>
</head>
<body>
  <table>
    <tr>
      <th>Data e ora</th>
      <th>Valore</th>
      {% for i in dizValori %}
        <tr>
          <td>{{i["DataOra"]}}</td>
          <td>{{i["Valore"]}}</td>
        </tr>
      {% endfor %}
    </tr>
  </table>
</body>
</html>

```

Nella seconda bisognava aggiungere i grafici utilizzando plotly e installando le due librerie *dash* e *pandas* installabili su Windows coi seguenti comandi:

```
py -m pip install panda
```

```
py -m pip install dash
```

codice Python plotly1:

```

import json
import plotly.express as px
import pandas as pd
import plotly
from flask import Flask, render_template
import os

path = os.getcwd()+'/datiSensore.json'
app = Flask(__name__)
@app.route('/')

def returnHtml():
    with open(path, 'r') as fp:
        lista = json.load(fp)

```

```

date = []
valoriSensori = []
for i in range(len(lista)):
    date.append(lista[i]["DataOra"])
    valoriSensori.append(lista[i]["Valore"])
df = pd.DataFrame({
    'DataOra': date,
    'Valore': valoriSensori})
fig = px.bar(df, x='DataOra', y='Valore', barmode='group')
graphJSON = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
return render_template('index.html', graphJSON=graphJSON)

```

template HTML:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv=refresh content=1>
    <title>Dati sensore plotly</title>
</head>
<body>
    <h1>Hello Plotly (but not Dash)</h1>
    <div id='chart' class='chart'></div>
</body><script src='https://cdn.plot.ly/plotly-latest.min.js'></script>
<script type='text/javascript'>
    var graphs = {{graphJSON | safe}};
    Plotly.plot('chart',graphs,{});
</script>
</html>

```

Nella terza parte abbiamo studiato il funzionamento dei decorators.