
ITSAR
SMART MANUFACTURING
• Front End
• Back End
• Model View Controller
Dott. Antonio Giovanni Lezzi

ITSAR

FLASK

- Flask è un micro-framework pronto per l'uso in produzione su una varietà di esigenze.
- Il "micro" in micro-framework significa che Flask mira a mantenere il core semplice ma estendibile.
- Flask è definito un framework "micro" perché non mette direttamente a disposizione funzionalità come la convalida del modulo, l'astrazione di database, l'autenticazione e così via.

ITSAR

FLASK

- Tali funzionalità sono invece contenute in speciali pacchetti Python denominati estensioni di Flask. Le estensioni si integrano perfettamente con Flask, come se facessero parte del framework.

- Flask permette di usare i componenti intercambiabili tra di loro, ad esempio se si pensa di utilizzare un tipo di database e poi si decide di cambiarlo con uno differente sarà possibile farlo

ITSAR

FLASK

- Flask si potrebbe considerare un vero e proprio web application server, cioè uno strumento con cui si possono gestire diverse applicazioni in grado di generare pagine web dinamiche facendo uso di un linguaggio di programmazione (in questo caso Python) invece che scrivere pagine HTML statiche.

- Questo permette di realizzare pagine web con contenuti variabili a seconda delle diverse scelte dell'utente e quindi la possibilità di creare siti professionali.

PYTHON: INSTALLAZIONE

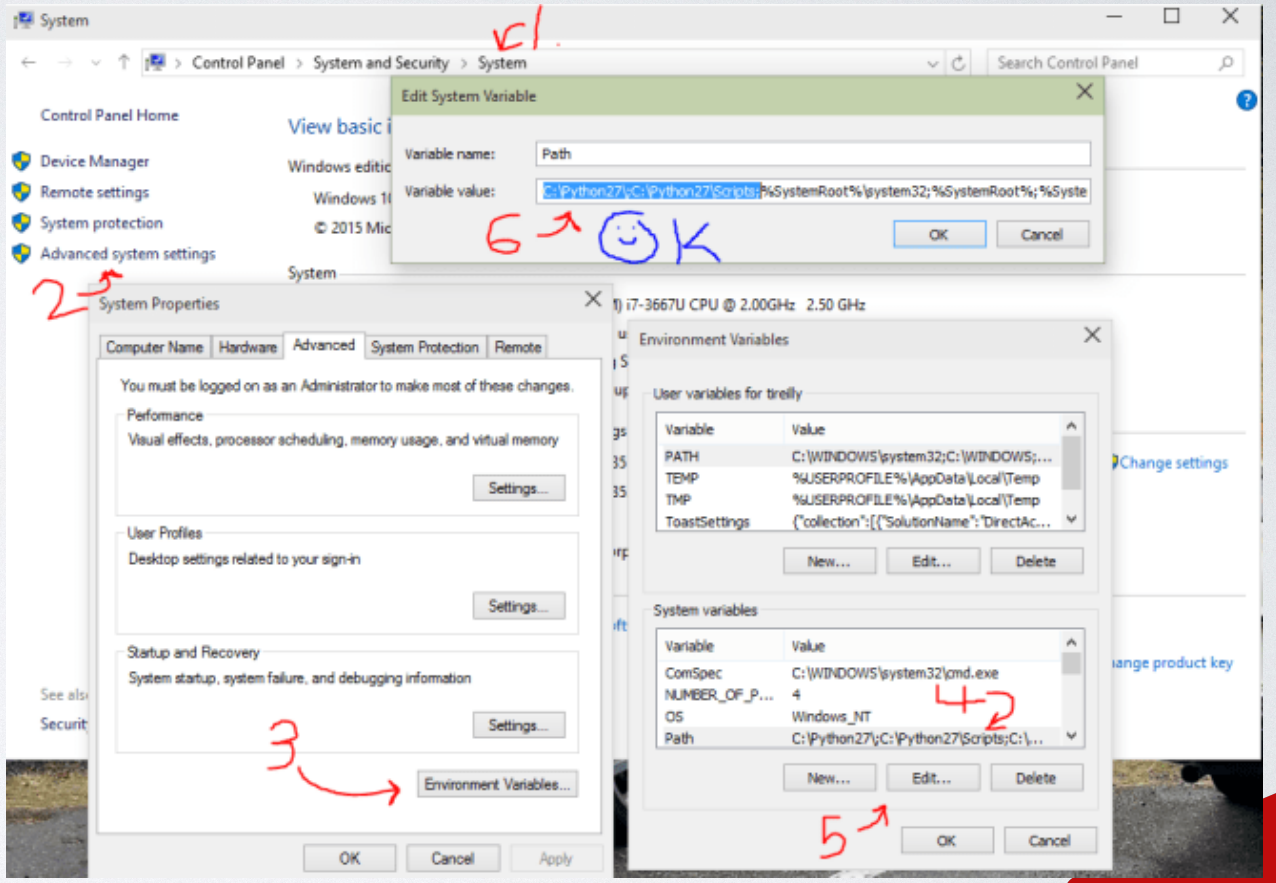
- Installare Python 3.7 e controllare siano stati selezionati gli elementi PIP e l'aggiunta del Path nelle variabili d'ambiente

- Per vedere le variabili d'ambiente si deve aprire il Pannello di Controllo -> Sistema e Sicurezza -> Sistema -> Impostazioni avanzate di Sistema -> Variabili d'Ambiente -> Selezionare Percorso

PYTHON: INSTALLAZIONE

- Se non dovesse esserci prestare molta attenzione alla modifica del Percorso, altrimenti si rischia anche di non far avviare più le applicazioni o addirittura il computer.

- Dovrebbero esserci I percorsi di dove si è installato Python come ad esempio C:\Python37; e C:\Python37\Scripts;.



FLASK: INSTALLAZIONE

- Procedere con l'apertura del prompt o del cmd e digitare
- ***pip install flask***

```
(HelloWold) C:\Users\tireilly\dev\HelloWold>pip install flask
Collecting flask
  Using cached Flask-0.10.1.tar.gz
Collecting Werkzeug>=0.7 (from flask)
  Downloading Werkzeug-0.10.4-py2.py3-none-any.whl (293kB)
    100% |################################| 294kB 220kB/s
Collecting Jinja2>=2.4 (from flask)
  Using cached Jinja2-2.7.3.tar.gz
Collecting itsdangerous>=0.21 (from flask)
  Using cached itsdangerous-0.24.tar.gz
Collecting markupsafe (from Jinja2>=2.4->flask)
  Using cached MarkupSafe-0.23.tar.gz
Installing collected packages: markupsafe, itsdangerous, Jinja2, Werkzeug, flask
  Running setup.py install for markupsafe
    building 'markupsafe._speedups' extension
    C:\Users\tireilly\AppData\Local\Programs\Common\Microsoft\Visual C++ for Python\9.0\V
C\Bin\amd64\cl.exe /c /nologo /Ox /MD /W3 /GS- /DNDEBUG -IC:\Python27\include -IC:\Users\
tireilly\Envs\HelloWold\PC /Tcmarkupsafe/_speedups.c /Fobuild\temp.win-amd64-2.7\Release\
markupsafe/_speedups.obj
    _speedups.c
    C:\Users\tireilly\AppData\Local\Programs\Common\Microsoft\Visual C++ for Python\9.0\V
C\Bin\amd64\link.exe /DLL /nologo /INCREMENTAL:NO /LIBPATH:C:\Python27\Libs /LIBPATH:C:\U
sers\tireilly\Envs\HelloWold\libs /LIBPATH:C:\Users\tireilly\Envs\HelloWold\PCbuild\amd64
 /EXPORT:init_speedups build\temp.win-amd64-2.7\Release\markupsafe/_speedups.obj /OUT:bui
ld\lib.win-amd64-2.7\markupsafe\_speedups.pyd /IMPLIB:build\temp.win-amd64-2.7\Release\ma
rkupsafe\_speedups.lib /MANIFESTFILE:build\temp.win-amd64-2.7\Release\markupsafe\_speedup
s.pyd.manifest
    _speedups.obj : warning LNK4197: export 'init_speedups' specified multiple times; usi
ng first specification
       Creating library build\temp.win-amd64-2.7\Release\markupsafe\_speedups.lib and obj
ect build\temp.win-amd64-2.7\Release\markupsafe\_speedups.exp
  Running setup.py install for itsdangerous
  Running setup.py install for Jinja2

  Running setup.py install for flask
Successfully installed Jinja2-2.7.3 Werkzeug-0.10.4 flask-0.10.1 itsdangerous-0.24 markup
safe-0.23

(HelloWold) C:\Users\tireilly\dev\HelloWold>
```

ITSAR

SCRIVERE CODICE PER FLASK

- Come creare un semplice server **Hello world** con **Flask**
- Ora che abbiamo Flask installato sul nostro PC, possiamo creare la nostra prima web app.
- In questo esempio andremo a creare un web server che ci mostrerà la scritta Hello world quando facciamo una richiesta all'URL **localhost:5000/**
- Come prima cosa create un nuovo file con estensione .py per esempio app.py

ITSAR

SCRIVERE CODICE PER FLASK

- All'interno di questo file andiamo ad inserire: http://localhost:5000/

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

SCRIVERE CODICE PER FLASK

Con questo codice stiamo:

- Importando Flask
- Creando una nuova app
- Definendo una route all'indirizzo home ('/') e collegarla alla funzione **hello()**
- Il metodo restituisce la parola Hello world al browser.

Possiamo eseguire il nostro server con il comando:

python app.py

Dal vostro browser navigate all'indirizzo **localhost:5000** e dovreste vedere la parola **Hello world** nel browser.

ITSAR

PAGINA HTML IN FLASK

- Una funzione che consenta di mostra una pagina HTML nel browser è la funzione **render_template**.
- Questa funzione prende come parametro, il nome della pagina HTML e la invia automaticamente al browser.
- Le pagine HTML da restituire devono esser cate all'interno di una cartella nella stessa directory del file **app.py** chiamata **templates**:

```
mkdir templates
```

PAGINA HTML IN FLASK

- Una volta creato e salvato il file HTML, possiamo modificare il codice Python usato nel passo precedente nel seguente modo:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/")
def hello():
    return render_template('index.html').
if __name__ == "__main__":
    app.run()
```

- Ora se riavviamo il server, possiamo vedere nel browser la pagina HTML che abbiamo creato. (per riavviare il server fare **Ctrl + c** e poi di nuovo **python app.py** nel terminal)

ITSAR

VARIABILI NEL CODICE HTML

- Possiamo utilizzare le variabili all'interno del nostro codice HTML per rendere la nostra pagina statica un po' più dinamica.
- La funzione **render_template()** accetta una serie di parametri che possiamo utilizzare per passare dei valori dal codice **Python** al codice **HTML**.

VARIABILI NEL CODICE HTML

- Per esempio se vogliamo passare il nome **Antonio** alla pagina HTML possiamo modificare il file **app.py** nel seguente modo:

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route(“/")
def hello():
    return render_template('index.html', nome="Antonio")
if __name__ == “__main__":
    app.run()
```

ITSAR

VARIABILI NEL CODICE HTML

- Come vedete nella funzione **render_template** passiamo una variabile chiamata nome con un valore di 'Antonio'.
- Per utilizzare questa variabile nella nostra pagina HTML, dobbiamo usare la notazione a doppia parentesi graffa. Per esempio:
- <h1>Ciao {{nome}}</h1>
- Se salvate il tutto e riavviate il server, dovreste vedere la scritta **Ciao Antonio** nel vostro browser all'indirizzo **localhost:5000**


ITSAR
SMART MANUFACTURING
Front End
Back End
Model View Controller
Dott. Antonio Giovanni Lezzi

MODEL VIEW CONTROLLER

- Il pattern architetturale usato è il Model-View-Controller (MVC) uno dei più diffusi nella programmazione object oriented e nello sviluppo di interfacce grafiche in quanto rappresenta uno dei concetti fondamentali della programmazione ad oggetti e permette di strutturare l'applicazione in maniera molto efficiente

- Aumenta la coesione del nostro sistema software, in quanto ogni singolo oggetto può ricoprire solo uno dei seguenti ruoli: modello, vista o controllore

- Infatti ogni classe del progetto viene mirata a compiti ben specifici, inoltre questi ruoli rappresentano una sorta di classificazione dell'oggetto che stiamo utilizzando che risultano dunque essere elementi logicamente separati ai quali però è consentita, ovviamente, una stretta comunicazione

ITSAR

MODEL VIEW CONTROLLER

- Il pattern non solo stabilisce che ruolo deve avere un determinato oggetto all'interno dell'applicazione, ma anche il modo in cui gli oggetti comunicano tra di loro
- **Model**: contiene i dati specifici dell'applicazione e si occupa di definire tutte le varie procedure che effettuano la manipolazione dei dati stessi, in lettura e scrittura
- Il modello non può avere connessione diretta con un oggetto di tipo view, in quanto ha il compito di gestire i dati che non devono essere legati ad un particolare tipo di visualizzazione

ITSAR

MODEL VIEW CONTROLLER

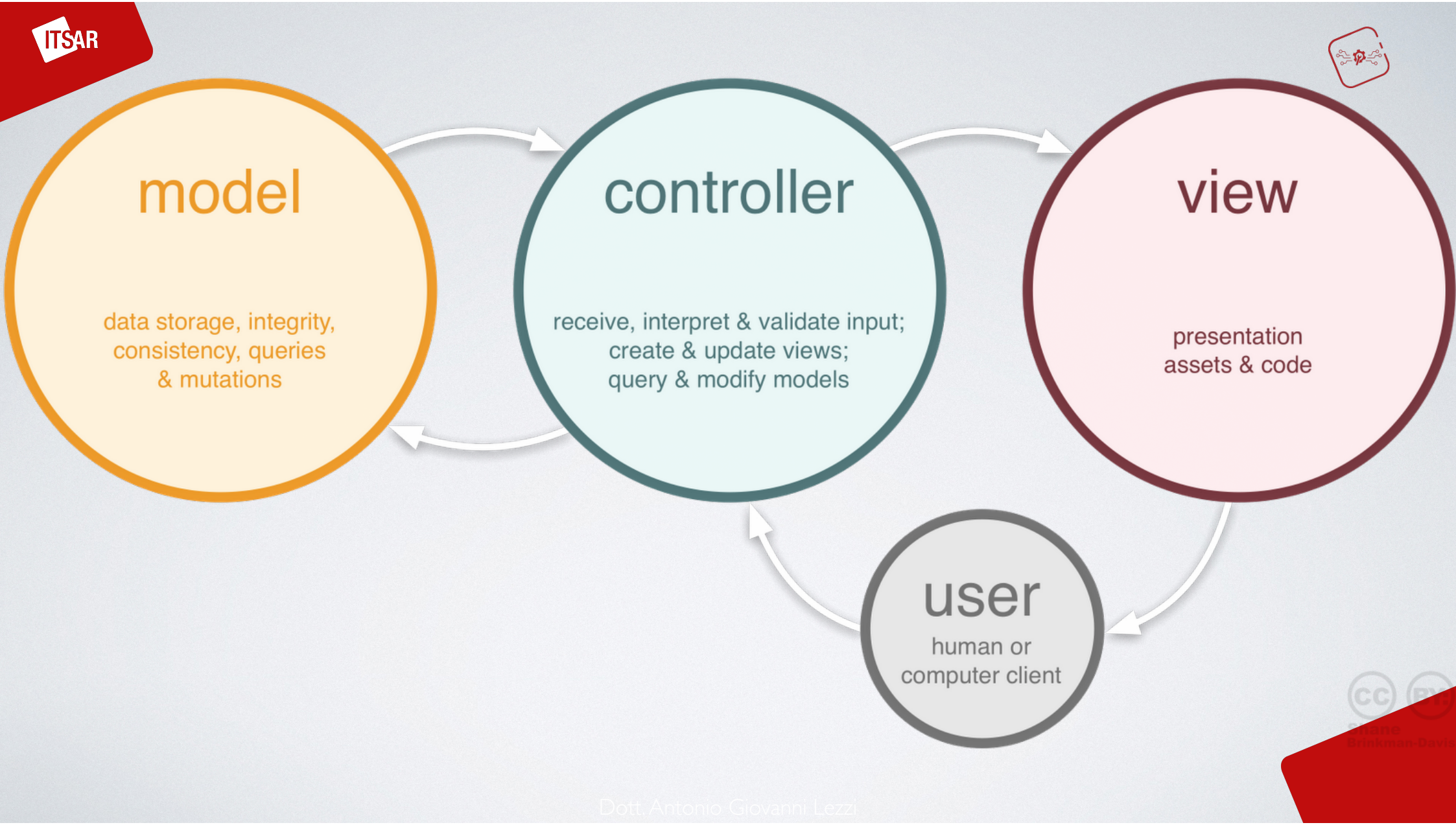
- **View**: il ruolo della vista è quello di presentare all'utente i dati contenuti all'interno di un modello. Concettualmente il modello è un oggetto non concreto, mentre la vista è un oggetto concreto e con il quale l'utente può interagire

- La vista è dunque una realizzazione di un oggetto non concreto e mette a disposizione un'interfaccia per la modifica dei dati contenuti nel modello

- L'oggetto di tipo vista non deve avere un riferimento esplicito ad un oggetto di tipo modello e quindi a questo punto viene usato l'oggetto controllore

MODEL VIEW CONTROLLER

- **Controller**: svolge la funzione di intermediario tra oggetti di tipo vista ed oggetti di tipo modello
- Un singolo controllore può avere un numero di relazioni arbitrarie tra oggetti di tipo modello e vista, che possono essere relazioni uno ad uno o molti a molti
- Il controllore si occupa di inizializzare la vista con i dati contenenti nel modello e informare alla vista le modifiche dei dati subite dall'utente


ITSAR
model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client