



SMART MANUFACTURING

Dott. Antonio Giovanni Lezzi





SMART MANUFACTURING

- **Introduzione IoT**
- Comunicazione Internet
- Servizi RESTful
- Client in Python

Dott. Antonio Giovanni Lezzi



INTRODUZIONE IOT

- **Internet of Things** (“internet delle cose”) è un neologismo nato dall’esigenza di dare un nome agli **oggetti reali e tecnologie connessi ad internet con qualsiasi altro apparato.**
- Internet of Things è rappresentata dalla **sensoristica** “semplice”, cioè **che rilevano Dati digitalizzati** e poterli **trasformarli in Informazioni** e poterli selezionare e **trasmettere solo quelli necessari** al progetto nel quale sono coinvolti.
- Questi Dispositivi sono in grado di gestire azioni sulla base delle indicazioni ricevute e di attuare azioni in funzione di una capacità elaborativa locale.



COMPONENTI UTILI NEL IOT

- IoT ha bisogno di **raccogliere e archiviare una grossa mole di dati, processare e analizzare grandi volumi di dati real time** (ad esempio dai sensori, dai semafori, e da qualsiasi dispositivo IoT connesso).
- Per questo c'è bisogno di sistemi integrati tra **big data, database nosql e dati IoT**.
- L'IoT apre prospettive in cui gli oggetti assumono un ruolo attivo grazie al fatto di essere in rete e di inviare e ricevere dati sulla rete e che acquisiscono intelligenza.



ESEMPI IOT

- IoT è ad esempio un frigorifero che ordina il latte quando “si accorge” che è finito.
- Una casa che accende i riscaldamenti appena ti sente arrivare.
- In ambito cittadino con un rilevatore collocato in una strada può controllare i lampioni e segnalare se la lampada funziona, ma lo stesso rilevatore potrebbe, se adeguatamente attrezzato, segnalare anche informazioni sulla qualità dell’aria o sulla presenza di persone.

SMART MANUFACTURING

Input article number

Display all referenced article numbers

Select one article number

Display all article information



Request

XML
SOAP
JSON

Introduzione IoT

Response

XML
SOAP
JSON

Request

XML
SOAP
JSON

Response

XML
SOAP
JSON

Comunicazione Internet

- Servizi RESTful
- Client in Python

Business logic

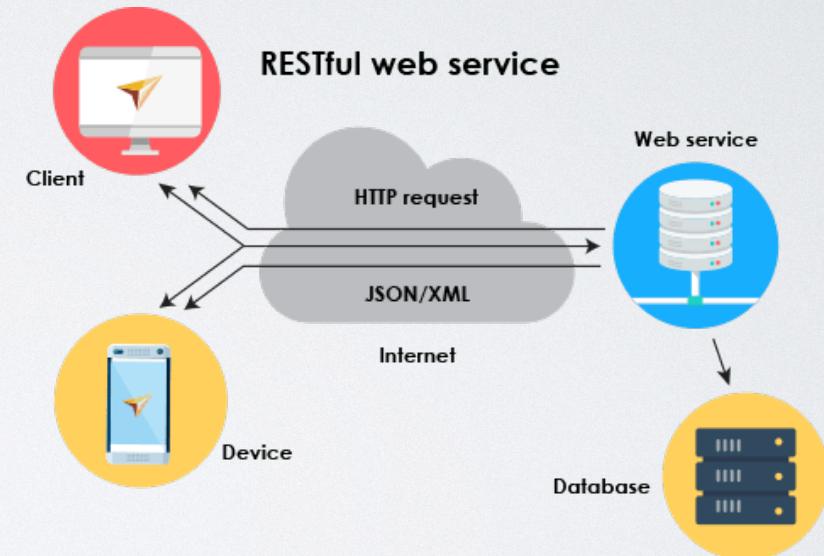


Dott. Antonio Giovanni Lezzi



COMUNICAZIONE INTERNET

- Ogni utente genera dati interagendo con vari dispositivi ed attraverso piattaforme di differente tipologia.
- La grande quantità di **informazioni** deve necessariamente **essere raccolta e memorizzata** in modo da poter essere **poi usata** immediatamente oppure in successivi momenti.
- L'acquisizione avviene attraverso API (interfaccia di programmazione di un'applicazione) utilizzate appositamente per raccogliere dati quando si accede ad un sito





COMUNICAZIONE INTERNET

- I computer in Internet comunicano scambiandosi **pacchetti di dati**, chiamati **pacchetti IP** (Internet Protocol)
- Questi pacchetti partono da un computer, **attraversano vari nodi** della rete (Server di rete) e arrivano a destinazione
- Per stabilire il percorso intermedio tra i due computer che vogliono comunicare si esegue un **algoritmo di routing** (ne esistono di vari tipi, a seconda delle esigenze e a seconda del tipo di rete).

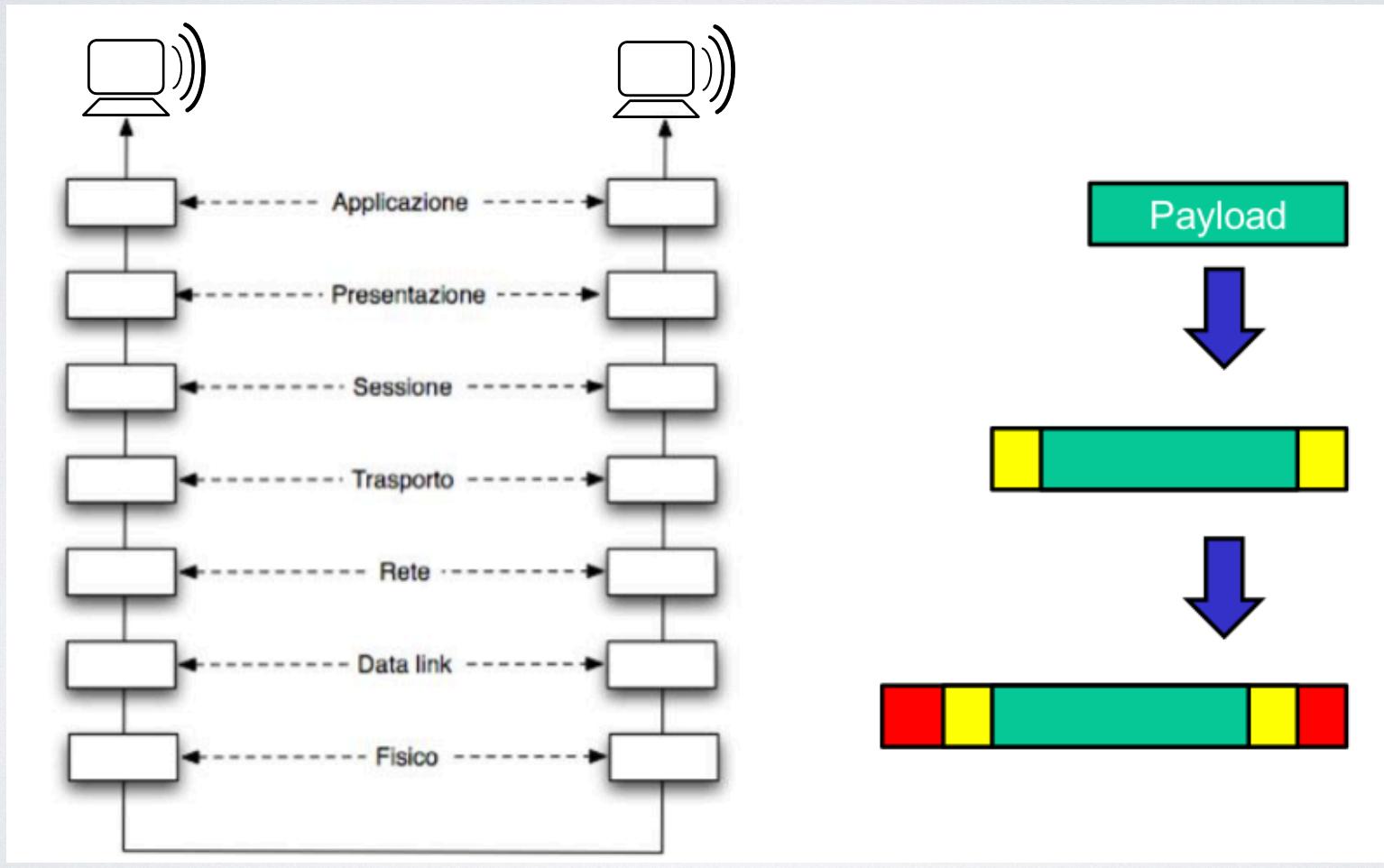


PARADIGMI DI COMUNICAZIONE

- **Client-Server:** Applicazioni di rete formate da due programmi distinti che possono essere in esecuzione in due elaboratori (host) diversi: un server e un client
 - **Server:** si mette in attesa di una richiesta da servire.
 - **Client:** effettua tale richiesta.
 - Tipicamente il client comunica con un solo server, mentre un server solitamente comunica con più client contemporaneamente.
- **Peer-to-peer:** Più host diversi comunicano tra loro comportandosi sia come client che come server.



COMUNICAZIONE A LIVELLI



Dott. Antonio Giovanni Lezzi



COMUNICAZIONE

- All'interno di una rete di calcolatori la comunicazione avviene tramite scambio di informazioni (messaggi)
- Per comunicare sono necessari:
 - **Canale fisico:** Cavo telefonico, fibra ottica, onde radio, ...
 - **Protocollo:** Insieme di regole formalmente descritte, definite al fine di favorire la comunicazione tra una o più entità.

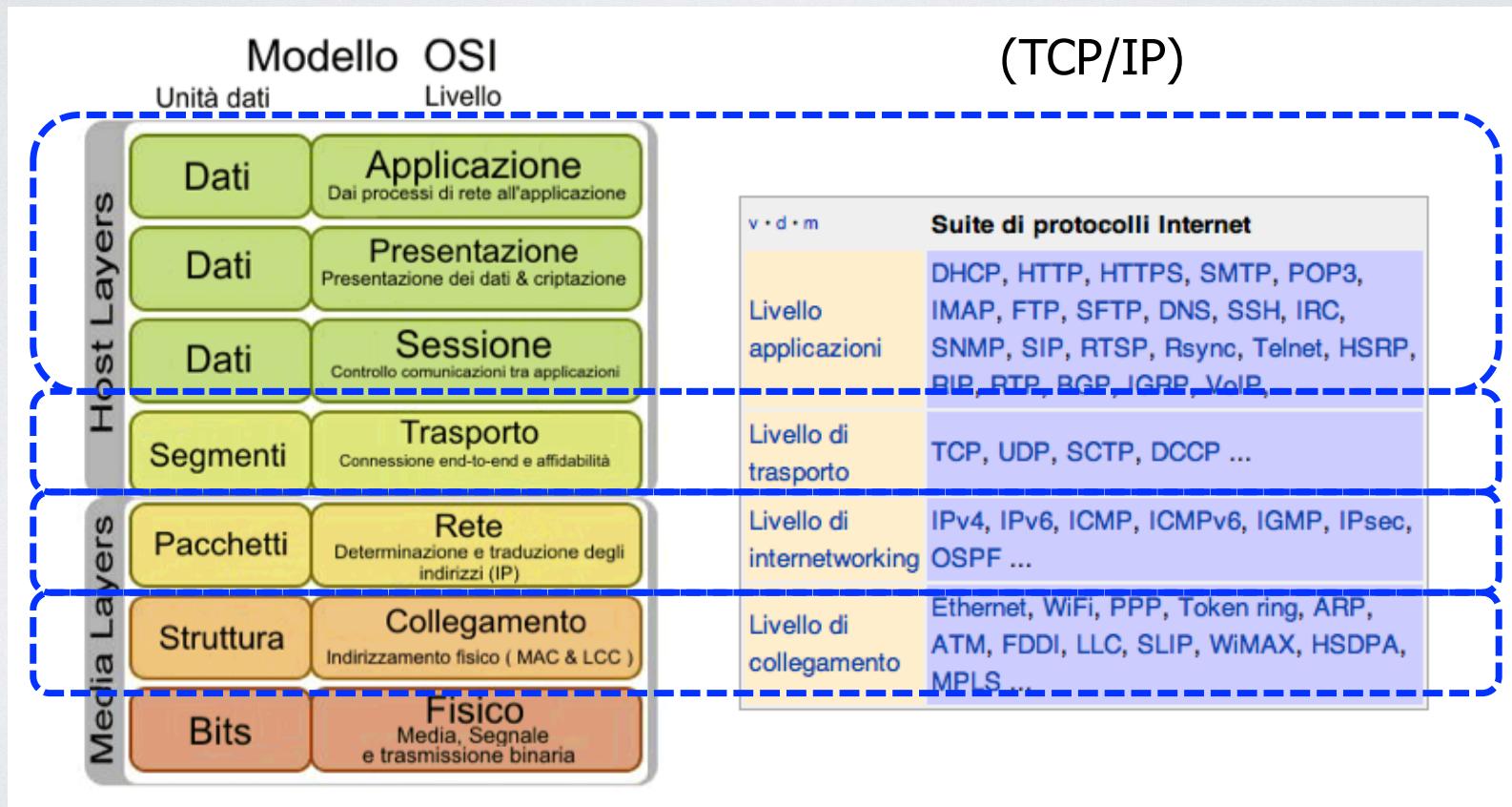


ALCUNI CONCETTI

- **Hosts:** sono i dispositivi connessi a Internet e sono computer, router, stampati, fax, macchina espresso, ecc
- **Internet Addresses:** il mittente di una comunicazione, i destinatari, i nodi intermedi, occorre che **ogni computer collegato** in rete abbia un nome che lo identifica univocamente, questo nome, è un numero, e si chiama **indirizzo IP**
- **Ports:** identifica un servizio da usare su un server



PILA PROTOCOLLARE



Dott. Antonio Giovanni Lezzi



ESEMPIO PROTOCOLLO

- Determinati protocolli sono gestiti da organismi quali il World Wide Web Consortium (W3C) che rispettano gli standard ISO/OSI, i protocolli più utilizzati:
 - **HTTP e HTTPS** - principale sistema per la trasmissione di informazioni sul web.
 - **Transmission Control Protocol (TCP)** è uno dei principali protocolli Internet su cui si appoggiano gran parte delle applicazioni web;
 - **User Datagram Protocol (UDP)** è usato di solito in combinazione con il protocollo IP.
 - **Internet Protocol (IP)** - un protocollo di rete a pacchetto;
- Esistono tuttavia anche protocolli non standardizzati ovvero di tipo proprietario, tipicamente adibiti a funzionalità di tecnologie altrettanto proprietarie.



TCP (TRANSFERT CONTROL PROTOCOL)

- È un protocollo di alto livello:
 - È orientato alla connessione;
 - Esegue controlli sugli errori, congestione e flusso di comunicazione
- Un destinatario TCP è identificato da un indirizzo IP (Internet Protocol) e da una porta di destinazione.
 - L'indirizzo IP identifica la macchina alla quale vogliamo collegarci.
 - Il numero della porta identifica l'applicativo (servizio) al quale vogliamo connetterci.
- Esempio: Un server web all'indirizzo 131.114.11.100 può offrire un servizio di posta elettronica e un servizio http. Il primo risponde sulla porta 110 e il secondo alla porta 80.



SMART MANUFACTURING

- Introduzione IoT
- Comunicazione Internet
- **Servizi RESTful**
- Client in Python

Dott. Antonio Giovanni Lezzi



BACKEND E FRONTEND

- Termini usati spesso in informatica e che, in base al contesto in cui sono inseriti, assumono diversi significati.
- Nel caso di un sito web:
- **front end** è la parte pubblica del sito (la parte del sito visibile all'utente e interpretato dal Browser)
- **back end** quella non accessibile dall'utente ma solamente agli sviluppatori del sito che, tramite la programmazione possono definire logiche di accesso, inserimento dati ecc.



BACKEND E FRONTEND

- Per fare una analogia con il ristorante potremmo dire che:
- Il frontend diventa quindi l'interfaccia attraverso cui ordinare il menu, ma anche la persona che ce lo consegna e prende eventuali altri ordini (scelta quindi delle forme, dei pulsanti, dei colori, dei font, della struttura grafica dei siti).
- Il backend è invece la cucina, che una volta ricevuto l'input iniziale, l'ordine appunto, prepara quanto richiesto e lo consegna tramite l'addetto, senza mai direttamente mostrarsi all'utente finale (capacità di comunicare con il server ed il database per restituire i risultati richiesti dagli utenti tramite l'interazione con la grafica).



URL

- **Uniform Resource Locator** è un metodo per **identificare univocamente la locazione della risorsa** su internet
- Una risorsa web può essere qualsiasi cosa, una directory, un file, un oggetto in rete come ad esempio una interfaccia per fare delle query ad un database remoto, o per un motore di ricerca.
- Tramite un URL è possibile riferirsi alle risorse di Internet in modo semplice e uniforme. Si ha così a disposizione una forma intelligente e pratica per identificare o indirizzare in modo univoco le informazioni su Internet.



URL

- Gli Uniform Resource Locator ci permettono di individuare in maniera univoca una risorsa nel Web, ed è composto dalle seguenti parti:
 - **protocollo**: normalmente http, https altri valori sono ftp, telnet, mailto;
 - **user e password (opzionali)**: sono le credenziali per accedere al Web Server;
 - **l'indirizzo del Web Server**: può essere indicato un IP address (157.138.20.15) o un nome (www.antonio.it);
 - **una porta (opzionale)**: indica il numero di porta TCP alla quale connettersi;
 - **il path della risorsa** (file normalmente) cercata;
 - **i parametri della richiesta (opzionali)**: importanti per le applicazioni Web permettono di scambiare utili parametri tra il browser e l'applicazione tramite ad esempio i FORM HTML.



URL

HTTP://ANTONIO:PASSWORD@WWW.SPORT.IT/SPORT/
PAG.HTML?CATEGORIA=CALCIO



MESSAGGI HTTP

- I **web server** permettono di ottenere **informazioni come risultato di una query** (interrogazione). Invece di richiedere un normale documento, si specifica nell'URL il nome di un programma, passandogli alcuni parametri che rappresentano la query vera e propria.
- **Una query ad un particolare servizio è costituita quindi da un normale URL, con in coda alcuni parametri.**
- La parte dell'URL che specifica i parametri inizia con un punto interrogativo (?). Ogni parametro è separato da una “e commerciale” (&), e i valori che si assegnano ai parametri sono specificati in questo modo: nome = valore (il valore è facoltativo).



MIME

- MIME è un acronimo di "**Multipurpose Internet Mail Extensions**".
- inizialmente utilizzato per inviare messaggi di email ma adottato in seguito anche per HTTP
- Quando un browser invia una richiesta al server web, si invia anche un header MIME



HEADER DI RICHIESTE HTTP

```
GET /javafaq/images/cup.gif HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/3.01 (Macintosh; I; PPC)
Host: www.oreilly.com:80
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
```



RISPOSTA DEL SERVER

- Quando un server web risponde a una richiesta di un browser un header ed un messaggio MIME viene inviato, ecco un esempio di risposta:

```
HTTP/1.0 200 OK
Server: Netscape-Enterprise/2.01
Date: Sat, 02 Aug 1997 07:52:46 GMT
Accept-ranges: bytes
Last-modified: Tue, 29 Jul 1997 15:06:46 GMT
Content-length: 2810
Content-type: text/html
```



STRUTTURA DEL WEB

Le applicazioni Web sono composte da 3 elementi:

- **URL**: i nomi delle risorse nel Web
- **HTTP**: il modo in cui vengono trasferite le risorse nel Web
- **HTML/JSON/XML**: il linguaggio delle risorse più importanti del Web



HTTP

- L'HyperText Transfer Protocol **stabilisce delle regole per il trasferimento delle informazioni**, il protocollo HTTP prevede le seguenti fasi:

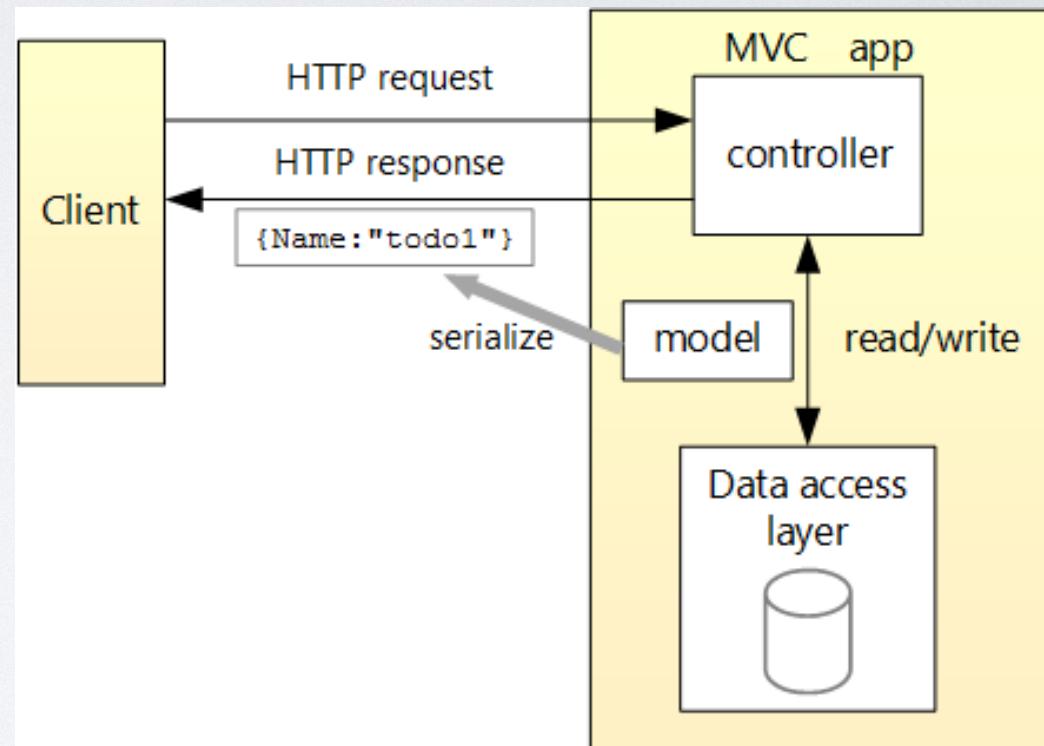
1. Il Browser o MobileApp apre una connessione col Web server specificato nel URL;
2. Si invia la richiesta al Web server attraverso la connessione;
3. il Web server invia la risposta al browser attraverso la connessione;
4. il server chiude la connessione.





SERVIZI WEB E RICHIESTE REST

- I servizi web sono essenzialmente funzionalità rese disponibili in rete da servizi remoti dai quali, client distribuiti nel mondo possono recuperare informazioni, richiedere elaborazioni e altro
- Le richieste sul protocollo HTTP possono esser di diverso tipo e per il servizio REST avremmo:
 - GET**: per leggere dati da remoto senza apportare modifiche
 - PUT**: per richiedere la modifica dei dati pre-esistenti
 - POST**: per inviare dati verso il servizio con lo scopo di richiederne l'inserimento nella base dati
 - DELETE**: per cancellare il dato





RICHIESTE REST

- POST, GET, PUT, DELETE richiamano gli stessi quattro concetti espressi dai metodi CRUD dei database (create, read, update e delete)
- in questo senso i servizi REST possono essere visti come un modo per gestire un database.



CODICI DI STATO

- i codici di stato contenuti nella risposta e forniti dal server dobbiamo sapere:
- **200**: OK
- **400**: Bad Request
- **403**: Forbidden
- **404**: Not found
- **500**: Internal Server error



RISPOSTA COMPLETA

GET javafaq/images/index.html HTTP/1.0

HTTP/1.1 302 Moved Temporarily

Date: Mon, 04 Aug 1997 14:21:27 GMT

Server: Apache/1.2b7

Location: <http://www.java.com/javafaq/images/index.html>

Connection: close

Content-type: text/html

<HTML><HEAD>

<TITLE>302 Moved Temporarily</TITLE>

</HEAD><BODY>

<H1>Moved Temporarily</H1>

The document has moved <A HREF="<http://www.macfaq.com/macfaq/index.html>">here.<P>

</BODY></HTML>



SMART MANUFACTURING

- Introduzione IoT
- Comunicazione Internet
- Servizi RESTful
- **Client in Python**

Dott. Antonio Giovanni Lezzi



RETE: RICHIESTE HTTP REST

```
#esempio semplice
import urllib.request

response = urllib.request.urlopen('http://www.google.org/')
html = response.read()

print(html)

#esempio con parametri
import urllib.parse
import urllib.request

url = 'http://www.server.com/register'
values = {'name' : 'Antonio Lezzi',
          'language' : 'Python' }

data = urllib.parse.urlencode(values)
print(data)
req = urllib.request.Request(url, data)
response = urllib.request.urlopen(req)
the_page = response.read()
print(the_page)
```



RETE: RICHIESTE HTTP REST

```
import urllib.parse
import urllib.request

url = 'http://www.server.com/register'
values = {'name' : 'Antonio Lezzi',
          'language' : 'Python' }
user_agent = 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'
headers = { 'User-Agent' : user_agent }

data = urllib.parse.urlencode(values)
req = urllib.request.Request(url, data, headers)
response = urllib.request.urlopen(req)
the_page = response.read()

print(the_page)
```



RETE: RICHIESTE HTTP REST

- La risposta ottenuta contiene tre parti:
 - Riga di stato
 - Header
 - Body
- La riga di stato consiste in un codice di tre cifre:
 - 1xx : Messaggi informativi
 - 2xx : La richiesta è stata effettuata con successo
 - 3xx : Non c'è risposta immediata, ma la richiesta è sensata
 - 4xx : La richiesta non può essere effettuata perché errata
 - 5xx : La richiesta non può essere effettuata per un errore nel server



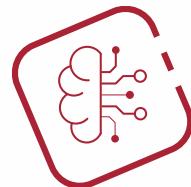
RETE: RICHIESTE HTTP REST

- Gli header più comuni sono due:
 - Server: indica tipo e versione del server
 - Content-Type: indica il tipo di contenuto restituito. Alcuni fra questi:
 - **1 text/html**
 - **2 text/plain**
 - **3 text/xml**
 - **4 image/jpeg**
- Il body contiene il contenuto della risposta.



WebTech & Cloud

Dott. Antonio Giovanni Lezzi





SMART MANUFACTURING

Dott. Antonio Giovanni Lezzi



SMART MANUFACTURING



- **Front End**
- Back End
- Model View Controller

Dott. Antonio Giovanni Lezzi



BROWSER

- **Il programma che interpreta il markup delle nostre pagine HTML e mostra a video tutto ciò che indichiamo si chiama Browser**
- Tra i compiti svolti dal browser, sono due quelli che ci interessano di più: richiesta di informazioni tramite URL, il caricamento e la visualizzazione (rendering) della pagina HTML



BROWSER

- I documenti HTML vengono immagazzinati sui server costantemente collegate e connesse alla rete.
- Su queste macchine Server è installato un software specifico (web server) che si occupa di produrre e inviare i documenti ai browser degli utenti che ne fanno richiesta usando il protocollo HTTP per il trasferimento dati.
- Quando ciò avviene la pagina richiesta viene elaborata dal browser dell'utente. Il risultato dell'elaborazione rappresenta la visualizzazione sullo schermo della pagina.



ELEMENTI IN HTML

- In una pagina HTML tutti gli elementi sono connotati da tag (letteralmente “etichette”)
- Si tratta di marcatori che evidenziano non tanto l’aspetto, quanto il senso, il ruolo, o l’organizzazione che vogliamo assegnare ai contenuti.
- Tuttavia in HTML possiamo anche definire lo scheletro dell’interfaccia utente di una app, e in questo frangente i tag diventano utili come supporto all’organizzazione del layout o alla definizione di aree specifiche per l’esperienza utente.
- In tutti e due i casi continuiamo a non parlare di “grafica”, ma di struttura.



TAG FONDAMENTALI

- **I TAG sono i comandi che definiscono la struttura di una pagina HTML** e che permettono una corretta visualizzazione da parte del browser
- **Sono racchiusi tra parentesi angolari ("<" e ">")** e possono essere digitati indifferentemente sia in minuscolo che in maiuscolo
- La maggior parte dei tag ha anche un tag correlato che ne indica il termine e si scrive anteponendo al nome del tag il simbolo slash “/”
- **i due tag formano una coppia e si chiamano tag di apertura e chiusura.**
Ad esempio, i tag che indicano l'inizio e la fine di un paragrafo si indicano con <P> e </P>



TAG FONDAMENTALI

- All'interno del tag possono essere inseriti degli attributi che delineano altre caratteristiche dell'elemento quali come la funzione, il colore, le dimensioni, la posizione relativa all'interno della pagina e i legami con il codice JavaScript.
- Un documento HTML è tutto ciò che è contenuto nei due **tag <HTML> e </HTML>**. Questi due tag sono generalmente il primo e l'ultimo di ogni documento HTML, e le prime versioni dei browser ignoravano qualunque cosa fosse all'esterno di questi tag
- **Un documento HTML si compone di due parti:** una intestazione e un corpo, in inglese rispettivamente **header e body**



Breakdown of an HTML Tag

Element

Attribute

Attribute
name
↓

Attribute
value
↓

```
<a href="hope.html">Computer Hope</a>
```

Opening tag

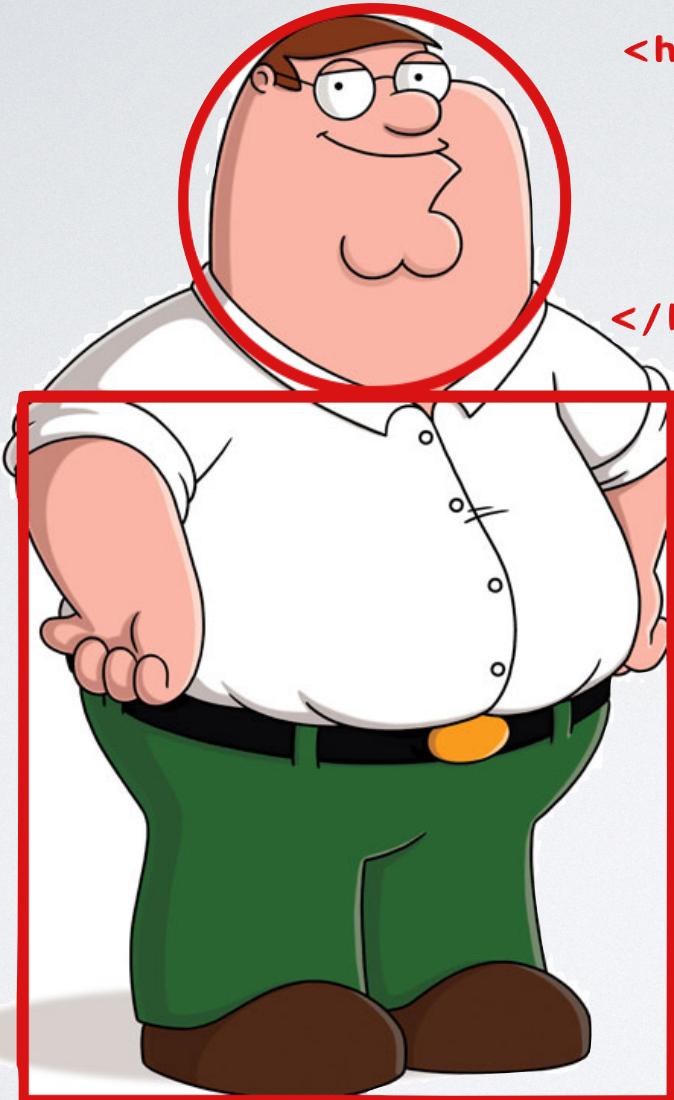
Enclosed tag content

Closing
tag



ANATOMIA DI UNA PAGINA

- L'header è un contenitore di informazioni, e racchiude tutte le specifiche, i parametri e le funzioni che non sono utili direttamente all'utente finale, ossia il lettore, ma che vengono utilizzate dai browser, dai motori di ricerca e da tutti quei programmi in grado di leggere un file HTML, per poter meglio gestire il documento
- Il body è la parte che effettivamente contiene tutte le cose che vengono visualizzate o che servono ad impostare la visualizzazione degli oggetti. Ancora una volta, i moderni browser si dimostrano abbastanza robusti ed elastici, visualizzando la pagina anche se la parte di header manca del tutto e se mancano i tag che delimitano la parte del body



<html>

<head>

<title>
Peter Griffin's HTML page
</title>

</head>

<body>

Page content goes here..

</body>

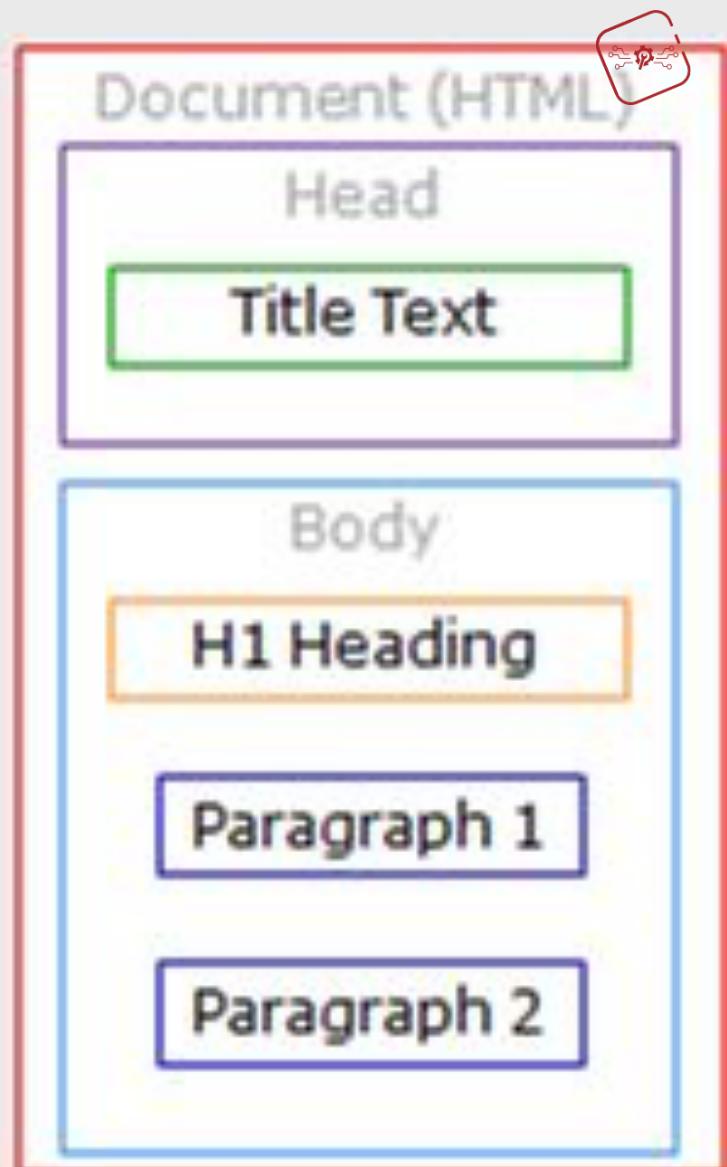
</html>

Dott. Antonio Giovanni Lezzi

```
<HTML>
  <HEAD>
    <TITLE>Title Text</TITLE>
  </HEAD>

  <BODY>
    <H1>H1 Heading</H1>
    <P>Paragraph 1</P>
    <P>Paragraph 2</P>
  </BODY>

</HTML>
```





FORM

- Un form normalmente è composto da vari tipi di controlli che consentono di inserire testo, selezionare delle opzioni ed effettuare altre operazioni disposte dal WebMaster
- L'utente conferma le sue scelte effettuando l'invio dei dati immessi, premendo l'apposito bottone a cui normalmente si da come etichetta Submit o Invia



FORM

- In seguito a questa operazione viene eseguito un programma residente sul server e a questo vengono passati i dati inseriti; il programma elabora i dati immessi ed effettua alcune operazioni per le quali è stato realizzato (ad esempio la visualizzazione di una pagina, l'invio di una E-Mail o altro).
- I programmi sono scritti solitamente in Python, Java, Perl o in C, ed utilizzano l'interfaccia CGI (Common Gateway Interface) per ricevere i dati dalla pagina web
- I metodi per passare i dati allo script sono sostanzialmente due: tramite una richiesta POST o tramite una richiesta GET; se non diversamente specificato è utilizzata la seconda possibilità



FORM

- Il tag che permette l'introduzione di un modulo è <FORM>, e tra esso ed il suo tag di chiusura può essere inserito qualunque contenuto HTML valido, tranne altri tag FORM
- Riconosce due attributi: ACTION="url" e METHOD="get | post"
- L'attributo ACTION è un po' come l'attributo SRC per le immagini; specifica l'indirizzo del programma che dovrà ricevere i dati immessi nel modulo
- METHOD specifica invece il modo in cui questi parametri vanno spediti al programma cgi, mediante l'url oppure mediante lo standard output



FORM: RACCOGLIERE INFO

- Il tag <INPUT> riceve alcuni attributi che indicano la natura della informazione introdotta
- L'attributo NAME="testo" specifica l'identificativo di quell'elemento introdotto, in pratica il nome della variabile a cui l'utente fornisce il valore.
- Infine l'attributo TYPE="tipo" che indica il tipo di informazione da raccogliere. Può assumere i valori
 - "text", "password"
 - "hidden"
 - "checkbox", "radio"
 - "submit", "reset", "image"



FORM: RACCOGLIERE INFO

```
<HTML>
  <HEAD>
    <TITLE>Manuale di HTML</TITLE>
  </HEAD>
  <BODY BGCOLOR="#BBFFBB">
    <H2 ALIGN="center">I moduli</H2>
    <P ALIGN="justify">Inserisci i dati richiesti. </P>
    <FORM ACTION="..url.." METHOD="post">
      Nome : <INPUT TYPE="text" NAME="nome" SIZE="18"><BR>
      Password : <INPUT TYPE="password" NAME="pwd" SIZE="15"><BR>
      Stato familiare<BR>
      single <INPUT TYPE="radio" NAME="sfam" VALUE="single" CHECKED>
      , coniugato <INPUT TYPE="radio" NAME="sfam" VALUE="coniugato"><BR>
      Hobbies<BR>
      pesca <INPUT TYPE="checkbox" NAME="pesca">, disegno <INPUT TYPE="checkbox" NAME="disegno"><BR>
      <INPUT TYPE="reset" VALUE="Annulla">
      <INPUT TYPE="submit" VALUE="Invia">
      <BR>
    </FORM>
  </BODY>
</HTML>
```



FORM: RACCOGLIERE INFO

I moduli

Inserisci i dati richiesti.

Nome :

Password :

Stato familiare
single , coniugato

Hobbies:
pesca , disegno



ESERCIZIO

- Definire delle pagine Html che consentano di effettuare:
- 1) l'autenticazione dell'utente richiedendo username e password
- 2) visualizzazione le informazioni dell'utente loggato (nome, cognome, città e altri dettagli)
- 3) pagina contenente un elenco dei prodotti (immagine e titolo)
- 4) pagina del dettaglio del prodotto (immagine, nome, prezzo, descrizione)
- Inoltre si richiede di definire un Database con la tabella Utenti(id, username, password, nome, cognome, città), tabella Prodotti(id, nome, prezzo, descrizione) e una Tabella Acquisiti(id, utenteid, prodottoid, quantità e data).
- Inserire alcuni dati nel database