

Metodologie per la Programmazione per il Web - MF0437

Esercizi - *Forms*

Docente

Giancarlo **Ruffo** [giancarlo.ruffo@uniupo.it]

Informazioni, materiale e risorse su:

moodle [<https://www.dir.uniupo.it/course/view.php?id=16455>]

- * Esercizi con form e spedizioni dati al server
- * Esercizi "Web Service"



- * **Esercizi con form e spedizioni dati al server**


- * Esercizi "Web Service"


Esercizio 1

- * Crea un file `.html` che consenta di inviare delle ricerche a Google

Basic Form

Type your Google query:





Nota1

- * Il file dell'esercizio precedente non necessita di alcuno script lato server
- * potrebbe essere richiamato anche senza passare da localhost

Form di base

```
<h1>Basic Form</h1>

<form action="http://www.google.com/search">
  <div>
    Type your Google query:
    <input name="q" />           <!-- text box -->
    <input type="submit" />      <!-- submit button -->
  </div>
</form>
```

Valori di default

- * Stesso esercizio di prima, ma passare un valore di default al form

Basic Form

Type your Google query:



Esercizio 2

- * Per l'esercizio seguente, usare lato server il codice (fornito via repo git) per mostrare su browser cosa riceve il server
- * andare via terminale nella cartella apposita
 - * `cd paramsapp`
- * eseguire il server da finestra terminale
 - * `node app`
- * Creare il form che vedete in figura ed inviare i dati al server con un comando **post (get)**:

Field Set: Registration

Login Information

username:

email:

password:

repeat your password :

Additional Information

full name:

birth date:

favourite donald's nephew: ☐ Huey ☐ Dewey ☒ Louie

favourite food:

your picture: No file selected.

Complete your Subscription

Verifica

- * Se tutto funziona correttamente, al submit vedrete questa pagina:

The screenshot shows a JSON viewer interface with a dark theme. At the top, there are tabs for 'JSON', 'Dati non elaborati', and 'Header'. Below the tabs are buttons for 'Salva', 'Copia', 'Comprimi tutto', 'Espandi tutto', and a search icon labeled 'Filtra JSON'. The JSON data is displayed in a tree structure. The root object has two main properties: 'params' and 'files'. The 'params' object contains fields for user registration: username, email, password1, password2, fullname, birthdate, ddnephew, and food. The 'files' object contains an array of file uploads, with the first file having fields for fieldname, originalname, encoding, mimetype, destination, filename, path, and size.

```
{
  "params": {
    "username": "gjaruffo",
    "email": "giancarlo.ruffo@uniupo.it",
    "password1": "ciao",
    "password2": "ciao",
    "fullname": "Giancarlo Ruffo",
    "birthdate": "1971-11-27",
    "ddnephew": "dewey",
    "food": "Pizza"
  },
  "files": [
    {
      "fieldname": "pic",
      "originalname": "profile pic.jpg",
      "encoding": "7bit",
      "mimetype": "image/jpeg",
      "destination": "./uploads",
      "filename": "1683804974989-profile pic.jpg",
      "path": "uploads/1683804974989-profile pic.jpg",
      "size": 153772
    }
  ]
}
```

Esercizio 3

- * Aggiungere all'esercizio precedente degli stili utili alle validazioni
- * vogliamo che l'utente capisca che tutti i campi del primo gruppo sono obbligatori

* Esercizi con form e spedizioni dati al server

* **Esercizio stile API REST**

Esercizio 4

- * Creare un file *power.html* che prende in input un valore numerico per *la base* ed un valore numerico per *l'esponente*
- * Spedire i dati al server *power.js* (fornito tramite repo git) tramite metodo *get*
- * Il file *power.js* deve essere eseguito con Nodejs e produce un output contenente il risultato
 - * in output viene mandato un oggetto *json* tramite la funzione *JSON.stringify*