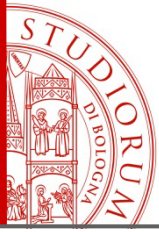


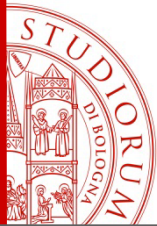
# Esercitazione PHP e AJAX



# Outline

---

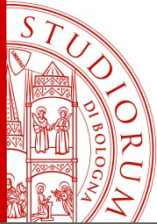
- Esercizi Php e Ajax



# Materiale a disposizione

---

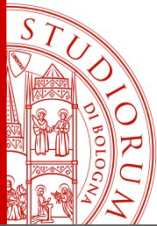
- Il materiale dell'esercitazione contiene una cartella con il codice dello scorso laboratorio (**con qualche aggiunta**). Nella cartella db trovate sempre gli script per creare e popolare il db.



# Esercizi

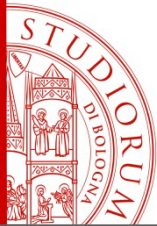
---

- Nel laboratorio precedente abbiamo creato tutte le pagine del sito.
- Oggi vedremo una versione alternativa delle pagine, usando Ajax. Il server restituirà le pagine con il tag main vuoto. Al caricamento della pagina, verrà effettuata una richiesta http per recuperare i dati dal server.
- Per le richieste Ajax, utilizzeremo la libreria Axios.



# A X I O S

- Axios è un client http basato su Promise per browser (e anche Node.js).
- Principali funzionalità:
  - Eseguire XMLHttpRequests dal browser
  - Basato sulle Promise  
([https://www.w3schools.com/js/js\\_promise.asp](https://www.w3schools.com/js/js_promise.asp))
  - Consente di intercettare richieste e risposte
  - Consente di cancellare le richieste Cancel requests
  - Trasformazione automatica dei dati in JSON
- Link alla documentazione ufficiale:  
<https://axios-http.com/>



# AXIOS

- Axios mette a disposizione un metodo per ogni metodo http.

`axios.request(config)`

`axios.get(url[, config])`

`axios.delete(url[, config])`

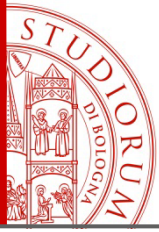
`axios.head(url[, config])`

`axios.options(url[, config])`

`axios.post(url[, data[, config]])`

`axios.put(url[, data[, config]])`

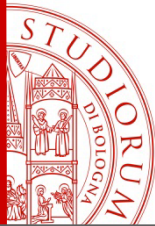
`axios.patch(url[, data[, config]])`



# AXIOS

- Esempio di richiesta GET

```
axios.get('risorsa-online')  
  .then(response => {  
    console.log(response);  
    //gestione della risposta  
  });
```

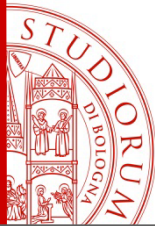


# Esercizio 01

---

- Popolare la homepage (creare un nuovo file `index2.php`) utilizzando Javascript, richiedendo i dati con Ajax:
  - `api-articolo.php` dovrà restituire un json contenente gli ultimi due articoli inseriti sul db.
  - `index2.php` non avrà un template specifico, ma restituirà il tag `main` vuoto.
  - `index.js` dovrà effettuare una richiesta `http get` alla pagina `api-articolo.php` e creare la struttura `html` utilizzando Javascript.





# Esercizio 02

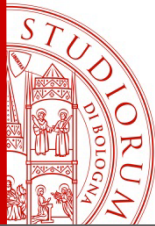
---

- Popolare la pagina contatti (creare un nuovo file contatti2.php) utilizzando Javascript, richiedendo i dati con Ajax:
  - api-contatti.php dovrà restituire un json contenente i dati degli autori presenti sul db.
  - contatti2.php non avrà un template specifico, ma restituirà il tag main vuoto.
  - contatti.js dovrà effettuare una richiesta http get alla pagina api-contatti.php e creare la struttura html utilizzando Javascript.



# Esercizio 03

- Gestire la pagina login (creare un nuovo file login2.php) utilizzando Javascript, richiedendo i dati con Ajax:
  - api-login.php dovrà:
    - in caso di richiesta get, restituire un oggetto json con proprietà logineseguito che vale true se l'utente è loggato, false altrimenti.
    - in caso di richiesta post, eseguire login, restituendo eventuali errori nel campo errorelogin.
    - in generale, se l'utente è loggato restituire nel campo articolautore gli articoli scritti dall'utente.
  - login2.php non avrà un template specifico, ma restituirà il tag main vuoto.



# Esercizio 03

– login.js dovrà:

- Al caricamento della pagina fare una richiesta get alla pagina api-login.
- Se il login è stato eseguito, visualizzare gli articoli.
- Altrimenti, visualizzare il form per il login e gestire il submit del form con Javascript. La richiesta Post deve essere effettuata con Axios (attenzione a come inviare i parametri in modo che siano leggibili da Php).
- In caso di errore, visualizzare il messaggio nel paragrafo.
- Altrimenti visualizzare gli articoli.