Laurea in Informatica

Programmazione e Sicurezza delle Reti
Introduzione alla programmazione web

Davide Tonin, Prof. Davide Quaglia

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Networked Systems and Technologies

Contatti

Davide Tonin
Email: davide.tonin@studenti.univr.it

Esercitazione

In questa esercitazione vedremo degli esempi di gestione degli eventi in javascript e utilizzo di api REST tramite richieste asincrone per inviare e recuperare i dati dal server e popolare la pagina web in modo dinamico.

In particolare:

- il primo esercizio prevede un form per effettuare il login (invio dei dati al server, validazione dei dati da parte del server, gestione della risposta da parte del client)
- il secondo esercizio richiede dei dati al server e popola in modo dinamico la pagina web (richiesta dei dati al server, gestione della risposta da parte del client)

Client/Server

Il client potrebbe essere una qualsiasi applicazione ma nel caso di questa esercitazione ci concentriamo sul browser. Questi i passaggi di una tipica connessione ad una pagina web:

- Il browser invia la richiesta;
- Il server elabora la risposta eseguendo script e programmi con un tipico linguaggio “lato server”: PHP, Java, Microsoft .NET, Python, Node.js e molti altri;
- Alla fine dell’esecuzione lato server, è stata prodotta una risposta composta da tutto ciò che un browser può interpretare ed eseguire: HTML, CSS, Javascript e risorse varie;
- Il server invia la risposta al browser che la interpreta e ne mostra i risultati all’utente. Il codice Javascript al suo interno rappresenterà l’intelligenza attiva di quella pagina, ciò che gestirà l’interazione utente.

Ogni volta che una pagina ha bisogno di richiedere elaborazioni o dati presenti sul server deve necessariamente avviare una nuova richiesta.

NOTA: ciò che Ajax ha portato di nuovo è che tali richieste possono essere eseguite in background ossia senza che l’utente si accorga di niente.

AJAX

AJAX, abbreviazione di **Asynchronous JavaScript and XML**, indica una combinazione di tecnologie di sviluppo usate per creare pagine web dal contenuto dinamico.

Nello specifico, la maggior parte delle implementazioni AJAX usa l'oggetto **XMLHttpRequest**, che include un elenco di funzioni di richiesta verso il server web che possono essere richiamate all'interno del codice JavaScript e quindi dentro la pagina web.

Ciò che caratterizza AJAX è che gli script vengono eseguiti lato client, consentendo di visualizzare immediatamente sulla pagina i dati ricevuti dal server, senza necessità di ricaricare la pagina stessa per visualizzarne i contenuti.

In questa esercitazione, useremo la libreria jQuery per inviare le richieste ajax al server.

JSON

JSON (JavaScript Object Notation) è un semplice formato di testo per lo scambio di dati, completamente indipendente dal linguaggio di programmazione, che utilizza convenzioni presenti in molti linguaggi e questo semplifica gli algosritmi di “parsing” automatico.

JSON è basato su due strutture:

- un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, record, struct, dizionario, array associativo,... a seconda del linguaggio utilizzato
- un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza come un array

Il formato di scambio dati viene definito nell’header della richiesta HTTP: **Accept: application/json**

Strumenti del Browser

I browser come Mozilla Firefox e Google Chrome mettono a disposizione una serie di strumenti utili per lo sviluppo web. Questi sono gli strumenti che andremo a vedere e provare:

- il primo pulsante a sinistra permette di selezionare un elemento della pagina caricata
- Analisi pagina: permette di visualizzare e modificare il codice html e css
- Console: contiene le informazioni di log associate alla pagina web (richieste di rete, errori/warning,...) ed altri messaggi registrati durante l'esecuzione del codice javascript. Permette inoltre di interagire con la pagina web eseguendo espressioni javascript nel contesto della pagina. (es. se nel codice javascript eseguo console.log("messaggio"), posso vedere questo messaggio nella console del browser.
- Archiviazione: permette di ispezionare vari tipi di archiviazione che la pagina web può utilizzare. Vedremo nelle prossime esercitazioni con We Plant.
- Rete: mostra tutte le richieste HTTP che vengono effettuate, con i relativi dettagli
- il terzo pulsante a partire da destra permette di testare la pagina web per dispositivi con diverse dimensioni (tablet/smartphone)

Pre-requisiti

Strumenti consigliati:

- Visual Studio Code
- Estensione Docker per Visual Studio Code (installabile direttamente dall'interno di visual studio code, nella sezione 'Extensions', cercando Docker).

Requisiti:

- docker

oppure

- python 3
- flask

Nel progetto è già presente un **Dockerfile** per lavorare con docker ed evitare di installare python direttamente sul computer.

Server

Comandi:

- docker build --tag server-es1 .
- docker run -d -p ***porta***:***porta*** server-es1

Qual è il valore di ***porta***? Su quale porta viene eseguito il server? (file: server.py)

Per vedere il log del server:

- docker ps (per vedere i container attivi)
- prendete l'id del container
- docker logs -f *id-container* (-f server per tenere aperto il log e vederlo in real-time)

A quale url risponde il server (dominio/ip e porta) ?

Esercizio 1

Modificare il file login.html:

- inserire un form con username e password (**ATTENZIONE:** entrambi i campi sono obbligatori, come lo gestisco?)
- aggiungere un flag per mostrare/nascondere la password
- gestire l'evento **onsubmit** sul form con la funzione ***login***
- gestire l'evento **onclick** sull'input per mostrare/nascondere la password con il metodo ***showPassword***

Modificare il file login.js e completare i metodi.

Esercizio 2

Modificare il file students.html:

- aggiungere una tabella o una lista per visualizzare gli elementi
- al momento del caricamento della pagina, la tabella o lista, è vuota o contiene degli elementi?
- aggiungere un pulsante che gestisce l'evento **onclick** e richiama il metodo **getStudents**, che carica i dati dal server e popola la pagina.

Modificare il file students.js e completare i metodi.