JQuery

Esercizio 1

Implementare un'applicazione Web lato client che visualizzi il risultato delle API comuni.php e popolazione.php sottoforma di tabella. In particolare, nel caso della comuni.php prevedere due pulsanti che permettano di scegliere se visualizzare le coordinate o l'url.

Soluzione

L'applicazione Web da implementare prevede tre componenti: il server (scritto in php), il database (in SQL) e il client (scritto in HTML+Javascript). Abbiamo già implementato il server e il database nelle esercitazioni precedenti, per cui ora ci concentreremo solo sulla parte client.
L'applicazione da implementare lato client prevede l'implementazione di una pagina html e di alcune funzioni javascript. Come prima cosa, dunque, nella cartella base del nostro progetto, creiamo una nuova cartella che chiameremo js e che conterrà tutti gli script javascript che andremo ad implementare. Dentro la cartella js creiamo un file che chiamiamo main.js. Per il momento lasciamo vuoto questo file. Ora torniamo nella cartella base del progetto, dove creiamo un file index.html, che conterrà le tabelle da visualizzare all'utente. Apriamo il file index.html e cominciamo a costruire un documento html di base.

```
<!DOCTYPE html>
<html lang="it">
      <head>
            <title>Comuni terremotati</title>
      </head>
      <body>
      </body>
</html>
```

Nell'intestazione (<head>) della pagina includiamo il lo script per l'utilizzo di jquery e il file main.js che si trova nella cartella js. Il file che contiene jquery deve essere incluso prima del file main.js, altrimenti all'interno di quest'ultimo non possiamo utilizzare le funzionalità di jquery.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>
<script src="js/main.js"></script>
```

A questo punto passiamo al corpo del file (identificato dal tag <body>). Dobbiamo prevedere un radio button che permetta di scegliere quale tabella visualizzare (comuni o popolazione) e una tabella che dovrà contenere i risultati.

```
<body>
      <div id="mio_div">
```

```
            <input type="radio" name="scelta" value="comuni"> Comuni
            <input type="radio" name="scelta" value="popolazione">
            Popolazione<br>
            <table id=”id_tabella”></table>
      </div>
</body>
```

Associamo alla tabella un id in modo da poterla recuperare in seguito dal javascript.
Passiamo ora al file main.js. Dobbiamo fare in modo che quando il DOM viene caricato, l’applicazione stia in ascolto dell’evento “cambio del pulsante”: se è selezionata l’opzione comuni, la pagina deve visualizzare i comuni, altrimenti la popolazione. Per questo motivo, utilizziamo la funzione ready() applicata al selettore $(document): il contenuto di questa funzione è eseguito una volta che il DOM è stato completamente caricato:

```
$( document ).ready(<parametri>);
```

Come unico parametro, la funzione ready() prende un’altra funzione. La sintassi completa è la seguente:

```
function mia_funzione()
{
      // corpo della funzione
}

$( document ).ready(mia_funzione);
```

In realtà, visto che la funzione mia_funzione è invocata solo come parametro della funzione ready, essa può essere inglobata nella funzione ready stessa. In questo caso, si può omettere il nome della funzione, che diventa pertanto una funzione anonima:

```
$( document ).ready(function(){
      //corpo della funzione
});
```

Passiamo ora al corpo della funzione. Abbiamo detto che dobbiamo fare in modo che al cambio della selezione da parte dell’utente sul radio button, deve cambiare anche la tabella visualizzata. Per fare questo, possiamo creare un ascoltatore sul radio button e ogni volta che questo cambia, cambiare il contenuto della tabella. In jquery per poter ascoltare un evento di cambiamento su qualcosa si può usare la funzione change applicata al radio button. Anche la funzione change riceve in ingresso un’altra funzione, che a sua volta riceve in ingresso l’evento che ha scaturito la chiamata alla funzione:

```
$("input[name=scelta]:radio").change(function(event)
{
      //corpo della funzione
});
```

All'interno di questa funzione, occorre eseguire operazioni diverse a seconda del tipo di evento che ha generato la chiamata. Preleviamo quindi il pulsante corrente e facciamo una discriminante sul valore:

```
var scelta = $("input[name=scelta]:checked").val();
if(scelta == 'popolazione')
      console.log( "popolazione" );
else
      console.log( "comuni" );
```

Il codice ottenuto fino ad ora è il seguente:

```
$( document ).ready(function(){
      $("input[name=scelta]:radio").change(function(event)
      {
            var scelta = $("input[name=scelta]:checked").val();
            if(scelta == "popolazione")
                  console.log( "popolazione" );
            else
                  console.log( "comuni" );
      });
});
```

Per testare la correttezza del nostro codice, apriamo il nostro file index.html dal browser (sempre digitando http://localhost seguito dal percorso della directory del nostro progetto). Una volta caricato il file, con il tasto destro del mouse clicchiamo sulla pagina e selezioniamo la voce ispeziona elemento. Scegliamo quindi la console. Proviamo ora a cambiare nella pagina da un tasto all'altro. Dovremmo vedere nella console comparire le scritte popolazione e comuni a seconda di ciò che pigiamo.
A questo punto possiamo passare all'implementazione del codice relativo alla popolazione. In questo caso dobbiamo chiamare l'API popolazione.php, leggere il risultato e formattarlo in una tabella. Per fare questo possiamo effettuare una chiamata AJAX mediante la funzione $.getJSON() che jquery mette a disposizione. Questa funzione può essere invocata solo se l'API restituisce un json. La funzione $.getJSON riceve in ingresso l'url a cui ci si vuole collegare e la funzione da eseguire in caso di successo. Come parametro di questa funzione è passato l'array data (che è popolato dalla getJSON e che contiene il risultato della chiamata all'API). Nel caso di popolazione possiamo passare alla getJSON una funzione crea_tabella, che poi utilizzermo anche nel caso dei comuni:

```
function crea_tabella(data){
      // stampa tabella
}
$.getJSON("api/popolazione.php", crea_tabella);
```

Ora passiamo a scrivere la funzione crea_tabella. In data abbiamo il risultato restituito dall'API popolazione.php. Ricordiamo che popolazione.php restituisce un array numerico, in cui ogni record è un array associativo. La tabella che vogliamo stampare dovrebbe essere di questo tipo:

Provincia	Popolazione
MC	82326
PG	12914

Per creare la riga di intestazione, dobbiamo appendere alla tabella identificata dall'id id_tabella una nuova riga. Usiamo la funzione append:

```
("#id_tabella").append("<tr id='id_intestazione_tabella'></tr>");
```

Ora dobbiamo creare le colonne dell'intestazione. Per fare questo, dobbiamo prendere il primo record dell'array numerico data per prelevare le chiavi (provincia e popolazione) dell'array associativo. Per scorrere l'array associativo, jquery mette a disposizione la funzione $.each, che riceve in ingresso l'array e una funzione a cui vengono passati la chiave e il valore:

```
$.each(obj, function(chiave, valore){
      // corpo each
});
```

Nel nostro caso passiamo alla funzione $.each() il primo record dell'array data e aggiungiamo alla riga identificata dall'id id_intestazione_tabella le colonne (<td>). Usiamo la funzione append() per appendere degli elementi al DOM:

```
$.each(data[0], function(chiave, valore){
      $("#id_intestazione_tabella").append("<td>" + chiave + "</td>");
});
```

Ora possiamo passare alle altre colonne. Possiamo scorrere nuovamente l'array data (attraverso un ciclo for) e questa volta prelevare i valori (attraverso la funzione $.each):

```
for(var i = 0; i < data.length; i++)
{
      ("#id_tabella").append("<tr id='id_riga" + i + "'></tr>");
      $.each(data[i], function(chiave, valore){
            $("#id_riga" + i).append("<td>" + valore + "</td>");

      });
}
```

Il codice completo del corpo della crea_tabella ottenuto fino ad ora è il seguente:

```
function crea_tabella(data)
{
      // stampa intestazione
      ("#id_tabella").append("<tr id='id_intestazione_tabella'></tr>");
      $.each(data[0], function(chiave, valore){
```

```
            $("#id_intestazione_tabella").append("<td>" + chiave +
            "</td>");
      });
      // stampa campi tabella
      for(var i = 0; i < data.length; i++)
      {
            ("#id_tabella").append("<tr id='id_riga" + i + "'></tr>");
            $.each(data[i], function(chiave, valore){
                  $("#id_riga" + i).append("<td>" + valore +
                  "</td>");
            });
      }
}
```

Ora passiamo alla comuni. In questo caso occorre discriminare il caso in cui si voglia ottenere le coordinate da quello in cui si vogliono le URL. In questo caso dobbiamo prevedere due pulsanti, uno per la scelta delle coordinate e uno per la scelta dell'url. Questi due bottoni devono essere creati dinamicamente quando si seleziona l'opzione comuni:

```
$( document ).ready(function(){
      $("input[name=scelta]:radio").change(function(event)
      {
            var scelta = $("input[name=scelta]:checked").val();
            if(scelta == "popolazione")
                  // codice popolazione
            else
            {
                  // creazione pulsanti
            }
      });
});
```

Creiamo due pulsanti da appendere al div identificato dall'id id_div:

```
$("#id_div").append('<button type="button" id="bURL">URL</button>');
$("#id_div").append('<button type="button"
id="bCoord">Coordinate</button>');
```

A questi due pulsanti dobbiamo associare degli ascoltatori sull'evento click. Al click di uno dei due pulsanti dovrà essere stampata la relativa tabella. Usiamo la funzione click per implementare l'ascoltatore:

```
$("#bURL").click(function(event){...});
$("#bCoord").click(function(event){...});
```

Il corpo delle due funzioni invocate dalle due click è abbastanza simile: ognuna contiene una chiamata ajax all'API comuni.php con parametro type diverso, a seconda del pulsante che ha generato l'evento. Nel caso della URL is ha:

```
$("#bURL").click(function(event){
      $.getJSON("api/comuni.php?type=url", crea_tabella);
});
```

Nel caso delle coordinate:

```
$("#bCoord)click(function(event){
      $.getJSON("api/comuni.php?type=coordinate", crea_tabella);
});
```

Il codice completo ottenuto fino ad ora è il seguente:

```
$( document ).ready(function(){
      $("input[name=scelta]:radio").change(function(event)
      {
            var scelta = $("input[name=scelta]:checked").val();
            if(scelta == "popolazione")
                  $.getJSON("api/popolazione.php", crea_tabella);
            else
            {
                  $("#id_div").append('<button type="button"
                  id="bURL">URL</button>');
                  $("#id_div").append('<button type="button"
                  id="bCoord">Coordinate</button>');

                  $("#bURL").click(function(event){
                        $.getJSON("api/comuni.php?type=url",
                        crea_tabella);
                  });

                  $("#bCoord").click(function(event){
                        $.getJSON("api/comuni.php?type=coordinate",
                        crea_tabella);
                  });
            }
      });
});
```

Ora resta da gestire un'ultima cosa. La prima è che ogni volta che si seleziona una opzione (comuni o popolazione) occorre rimuovere il contenuto della precedente selezione. Per fare questo, possiamo definire una funzione cancella da invocare prima della creazione della nuova tabella. In questa funzione invochiamo la funzione empty applicata al div della tabella:

```
function cancella()
{
      (“#id_tabella”).empty();
}
```

Inoltre dobbiamo eliminare i pulsanti creati all’inizio della comuni. Per fare questo chiamiamo la remove sui pulsanti. Rispetto alla empty, che cancella il contenuto di un elemento, la remove rimuove l’intero elemento. Verifichiamo prima che i pulsanti esistano e in caso affermativo li cancelliamo:

```
function cancella()
{
      (“#id_tabella”).empty();
      if ( $( "#bURL" ).length )
            $(“#bURL”).remove();
      if ( $( "#bCoord" ).length )
            $(“#bCoord”).remove();
}
```

A questo punto possiamo chiamare la cancella pulsanti prima di eseguire le operazioni sia della comuni che della popolazione:

```
            if(scelta == “popolazione”)
            {
                  cancella();
                  $.getJSON(“api/popolazione.php”, crea_tabella);
            }
            else
            {
                  cancella();
                  $(“#id_div”).append(‘<button type="button"
                  id=”bURL”>URL</button>’);
                  $(“#id_div”).append(‘<button type="button"
                  id=”bCoord”>Coordinate</button>’);

                  $(“#bURL”).click(function(event){
                        $.getJSON(“api/comuni.php?type=url”,
                        crea_tabella);
                  });

                  $(“#bCoord”).click(function(event){
                        $.getJSON(“api/comuni.php?type=coordinate”,
                        crea_tabella);
                  });
            }
```

Esercizio 2

Riorganizzare il codice della $(document).ready(function(){...}); in modo ridurre la replicazione delle istruzioni (usare le funzioni).