Funzioni

Esercizio 1

Organizzare il codice delle api comuni.php e popolazioni.php in modo da ridurre la ridondanza del codice.

Soluzione

Per ridurre la ridondanza del codice, occorre utilizzare le funzioni, che possono essere utilizzate per raggruppare spezzoni di codice che sono usati più di una volta. Quindi una buona norma di programmazione è la seguente: quando ci si ritrova a scrivere lo stesso pezzo di codice due volte, conviene raggruppare il codice attraverso l'uso delle funzioni.
Per prima cosa, nella nostra cartella del progetto creiamo una cartella php, in cui metteremo tutto il codice php che non costituisce le API. All'interno di questa cartella, creiamo un file funzioni.php, che conterrà le funzioni che via via definiremo.
Nel nostro caso il file comuni.php (con la GET) e popolazioni.php condividono delle operazioni:

1. eseguono una query
2. scorrono l'array del risultato e modificano un campo che corrisponde ad una certa chiave

Possiamo quindi creare una funzione che effettua queste operazioni e che poi è invocata dai due script.
Apriamo il file funzioni.php e cominciamo a scrivere. La funzione riceve in ingresso la query, che è il parametro che cambia nei due script:

```
<?php
function get_risultati($sql)
{
      // corpo della funzione
}
?>
```

Questa funzione deve effettuare le due operazioni precedenti. Tuttavia per poter effettuare la query, deve invocare la funzione select, che riceve in ingresso anche la variabile di connessione al db. Possiamo adottare due strategie differenti. O passiamo la variabile $db come parametro della funzione, oppure invochiamo la config.php dentro la funzione. Optiamo per questa seconda opzione, cioè invochiamo la config.php dentro la funzione. Il file config.php si trova dentro la cartella api per cui in teoria come percorso dovremmo mettere ../api/config.php. Tuttavia, poiché le funzioni sono invocate dentro la api, è sufficiente omettere il percorso.

```
<?php
function get_risultati($sql)
{
      include('config.php');
}
?>
```

A questo punto possiamo copiare il codice di uno dei due script (ad esempio comuni.php dentro la funzione:

```
<?php
function get_risultati($sql)
{
      include('config.php');
      $risultato = select($db,$sql);
      for($i = 0; $i < count($risultato); $i++)
      {
            foreach($risultato[$i] as $chiave => $valore)
            {
                  if(($chiave == 'latitude') ||
                        ($chiave == 'longitude'))
                        $risultato[$i][$chiave] = floatval($valore);
            }
      }
      closeDB($db);
      return $risultato;
}
?>
```

Come ultimo statement della funzione abbiamo messo la return, che permette di restituire il risultato. Questo codice però non funziona correttamente nel caso di popolazione, in quanto in questo caso occorre verificare la chiave 'popolazione' e non latitude e longitude. Possiamo quindi trasformare l'if in uno switch, in cui includiamo la verifica della popolazione:

```
switch($chiave)
{
      case 'latitude': case 'longitude':
            $risultato[$i][$chiave] = floatval($valore);
            break;
      case 'Popolazione':
            $risultato[$i][$chiave] = intval($valore);
            break;
}
```

Il codice completo della funzione è il seguente:

```
<?php
function get_risultati($sql)
{
      include('config.php');
      $risultato = select($db,$sql);
      for($i = 0; $i < count($risultato); $i++)
      {
```

```
            foreach($risultato[$i] as $chiave => $valore)
            {
                  switch($chiave)
                  {
                        case 'latitude': case 'longitude':
                              $risultato[$i][$chiave] = floatval($valore);
                              break;
                        case 'Popolazione':
                              $risultato[$i][$chiave] = intval($valore);
                              break;
                  }
            }
      }
      closeDB($db);
      return $risultato;
}
?>
```

Ora passiamo a sistemare i due script comuni.php e popolazione.php. Partiamo da comuni.php. La prima cosa da fare è includere, all'inizio dello script, il nostro file funzioni.php.

```
include('../php/funzioni.php');
```

Poi, sostituiamo tutto il codice che abbiamo messo dentro la funzione con la chiamata alla funzione. Il codice completo di comuni.php è il seguente:

```
<?php

header('Content-Type: application/json');
include('../php/funzioni.php');

if(isset($_GET['type']))
{
      $type = $_GET['type'];
      $sql = "";
      switch($type)
      {
            case 'coordinate':
                  $sql = "SELECT Comune, latitude, longitude FROM
ComuniTerremotati";
                  break;
            case 'url':
                  $sql = "SELECT Comune,url FROM ComuniTerremotati";
                  break;
            default:
                  echo json_encode(array("tipo non riconosciuto"));
                  exit(1);
      }
```

```
	$risultato = get_risultati($sql);
	echo json_encode($risultato);
}
else
{
	echo json_encode(array("parametro mancante"));
}
?>
```

Eseguiamo la stessa operazione per popolazioni.php

```
<?php

header('Content-Type: application/json');
include('../php/funzioni.php');
$sql = "SELECT Provincia, SUM(Popolazione) AS Popolazione FROM ComuniTerremotati GROUP BY Provincia";
$risultato = get_risultati($sql);
echo json_encode($risultato);

?>
```

Guardando il codice e volendo ulteriormente estremizzarlo, possiamo definire una funzione che costruisce la stringa contenente la query. La funzione potrebbe ricevere in ingresso i seguenti parametri:

- un array numerico contenente i campi della select
- una stringa contenente il nome della tabella
- un array associativo contenente i campi della where
- una stringa contenente le condizioni opzionali (ad esempio la GROUP BY)

Iniziamo con il definire la funzione:

```
function costruisci_select($select,$from, $where, $optional)
{
	// corpo della funzione
}
```

Creiamo una variabile $sql a cui man mano appenderemo il risultato a cui inizialmente assegniamo la stringa SELECT seguita da uno spazio:

```
$sql = "SELECT ";
```

Analizziamo ora la select. Dobbiamo scorrere l'array numerico e appendere alla variabile $sql gli elementi dell'array, seguiti dal carattere virgola e da uno spazio.

```
for($i = 0; $i < count($select); $i++)
	$sql .= $select[$i].", ";
```

Alla fine di questo for la variabile $sql dovrebbe contenere una stringa di questo tipo:

SELECT campo1, campo2, campo3,

Come possiamo vedere dall'esempio, abbiamo aggiunto una virgola e uno spazio in più, che possiamo togliere fuori dal ciclo for attraverso la funzione substr, che permette di ricavare una sottostringa da una stringa.
La funzione substr riceve in ingresso la stringa da modificare, l'offset a partire dal quale si vuole ottenere la sottostringa e la lunghezza della sottostringa. Restituisce la sottostringa cercata:

```
substr($stringa_di_partenza, $offset_iniziale, $lunghezza_sottostringa)
```

Nel nostro caso noi vogliamo una stringa che contenga tutti i caratteri della stringa $sql, ad eccezione degli ultimi due, che sono la virgola e lo spazio. Possiamo ottenere la lunghezza della stringa attraverso la funzione count, a cui però dobbiamo sottrarre 3, per eliminare la virgola e lo spazio. La lunghezza della stringa va da zero a count(stringa) -1, per cui oltre a togliere i due caratteri (la virgola e lo spazio) occorre togliere anche 1. Assegniamo il risultato nuovamente alla variabile $sql:

```
$sql = substr($sql, 0, count($sql) - 2);
```

Il codice ottenuto per la select è il seguente:

```
function costruisci_select($select,$from, $where, $optional)
{
      for($i = 0; $i < count($select); $i++)
            $sql .= $select[$i].", ";
      $sql = substr($sql, 0, count($sql) - 2);

}
```

Passiamo ora alla from. Dobbiamo solo aggiungere alla variabile $sql la parola chiave FROM preceduta e seguita da uno spazio e il nome della tabella, contenuta nella variabile $from:

```
$sql .= " FROM ".$from;
```

Passiamo infine alla clausola WHERE. L'array $where è un array associativo del tipo: nome_del_campo => valore. Come prima cosa appendiamo alla variabile $sql la parola chiave WHERE, preceduta e seguita da uno spazio:

```
$sql .= " WHERE ";
```

Poi scorriamo l'array associativo $where attraverso il costrutto foreach e appendiamo alla variabile $sql stringhe del tipo nome_campo = 'valore_campo', aggiungendo alla fine di ogni stringa la parola AND seguita da uno spazio:

```
foreach($where as $chiave => $valore)
{
      $sql .= $chiave = "'$valore' AND ";
}
```

Come nel caso della select, anche in questo caso la stringa finale avrà una AND in più, che può essere rimossa attraverso l'uso della funzione substr:

```
$sql = substr($sq, 0, count($sql) - count("AND ")-1);
```

Abbiamo usato la funzione count per calcolare la lunghezza della stringa "AND ".
Nel caso della where dobbiamo anche gestire il caso in cui la where non è prevista. Assumiamo che la funzione riceva in ingresso null se non si vuole la where. Dobbiamo quindi fare un controllo che la variabile $where sia diversa da zero:

```
if(!is_null($where))
{
      $sql .= "WHERE ";
      foreach($where as $chiave => $valore)
      {
            $sql .= $chiave = "'$valore' AND ";
      }
      $sql = substr($sq, 0, count($sql) - count("AND ")-1);
}
```

Abbiamo usato la funzione is_null per verificare che la variabile sia nulla, abbiamo poi aggiunto l'operatore ! per verificare che la variabile non sia nulla.
Ora, come ultima operazione possiamo appendere alla variabile $sql la stringa $optional e restituire il valore dell'intera stringa:

```
$sql .= $optional;
return $sql;
```

Il codice completo della funzione è il seguente:

```
function costruisci_select($select,$from, $where, $optional)
{
      $sql = "SELECT ";
      for($i = 0; $i < count($select); $i++)
            $sql .= $select[$i].", ";
      $sql = substr($sql, 0, count($sql) - 3);
      $sql .= " FROM ".$from;
      if(!is_null($where))
      {

            $sql .= " WHERE ";
            foreach($where as $chiave => $valore)
            {
                  $sql .= $chiave = "'$valore' AND ";
            }
            $sql = substr($sq, 0, count($sql) - count("AND "));
      }
```

```
    $sql .= " ".$optional;
    return $sql;
}
```

Ora possiamo invocare la funzione costruisci_select sia da comuni.php che da popolazioni.php. Nel caso di comuni.php si hanno due chiamate alla funzione costruisci_select.

Nel caso delle coordinate:

```
$sql = costruisci_select(array('Comune', 'latitude', 'longitude')
,'ComuniTerremotati', null, ''));
```

Nel caso dell'url:

```
$sql = costruisci_select(array('Comune', 'url') ,'ComuniTerremotati',
null, ''));
```

Nel caso dello script popolazioni.php, la chiamata alla funzione diventa la seguente:

```
$sql = costruisci_select(array('Provincia', 'SUM(Popolazione) AS
Popolazione') ,'ComuniTerremotati', null, 'GROUP BY Provincia'));
```

Esercizio 2

Modificare la funzione get_risultati in modo da inglobare la json_encode.