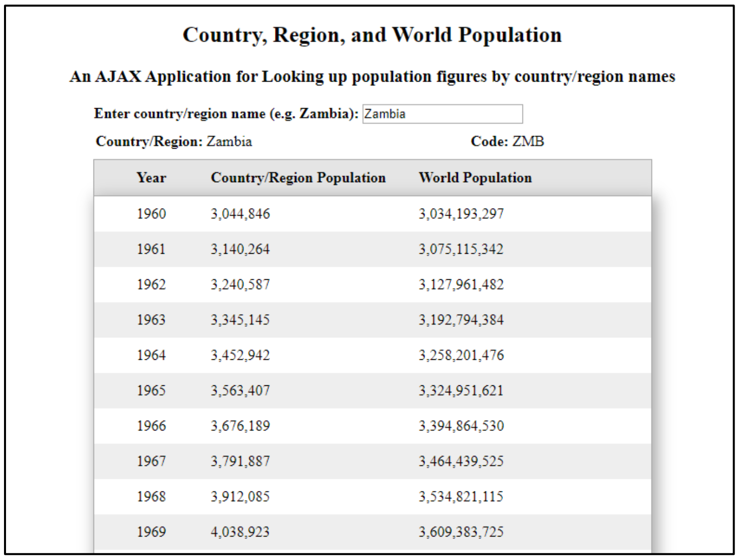
Lab 5: Country, Region, and World Population

How the Application Works

This application allows a user to look up population by a country/region name. When a valid name is entered into the search box, the population of the named country and the whole world displays; if the name is not valid, the message "Name not found" displays next to the search box. See screen shots below for sample outputs. Since AJAX is enabled in the application, there is no button to click; there is no need to press the **enter** key to submit the name. The name is submitted to the server by a browser on behalf of the user and the browser updates the Web page when server's response arrives. Please try the complete application at https://i211.sitehost.iu.edu.

A valid name is entered. The table shows population of the named country and the world.



Country, Region, and World Population

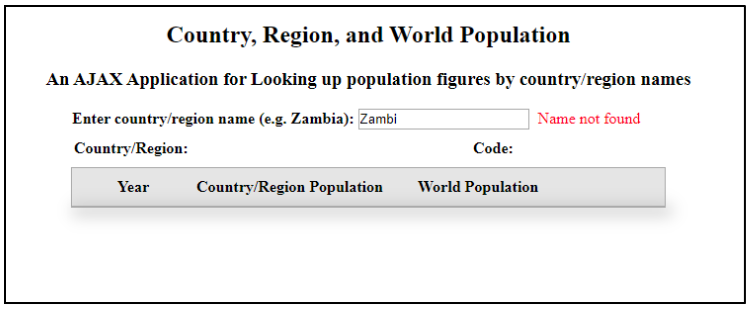
An AJAX Application for Looking up population figures by country/region names

Enter country/region name (e.g. Zambia): Zambia

Country/Region: Zambia **Code:** ZMB

Year	Country/Region Population	World Population
1960	3,044,846	3,034,193,297
1961	3,140,264	3,075,115,342
1962	3,240,587	3,127,961,482
1963	3,345,145	3,192,794,384
1964	3,452,942	3,258,201,476
1965	3,563,407	3,324,951,621
1966	3,676,189	3,394,864,530
1967	3,791,887	3,464,439,525
1968	3,912,085	3,534,821,115
1969	4,038,923	3,609,383,725

An invalid name is entered. A red message shows the name cannot be found.



Country, Region, and World Population

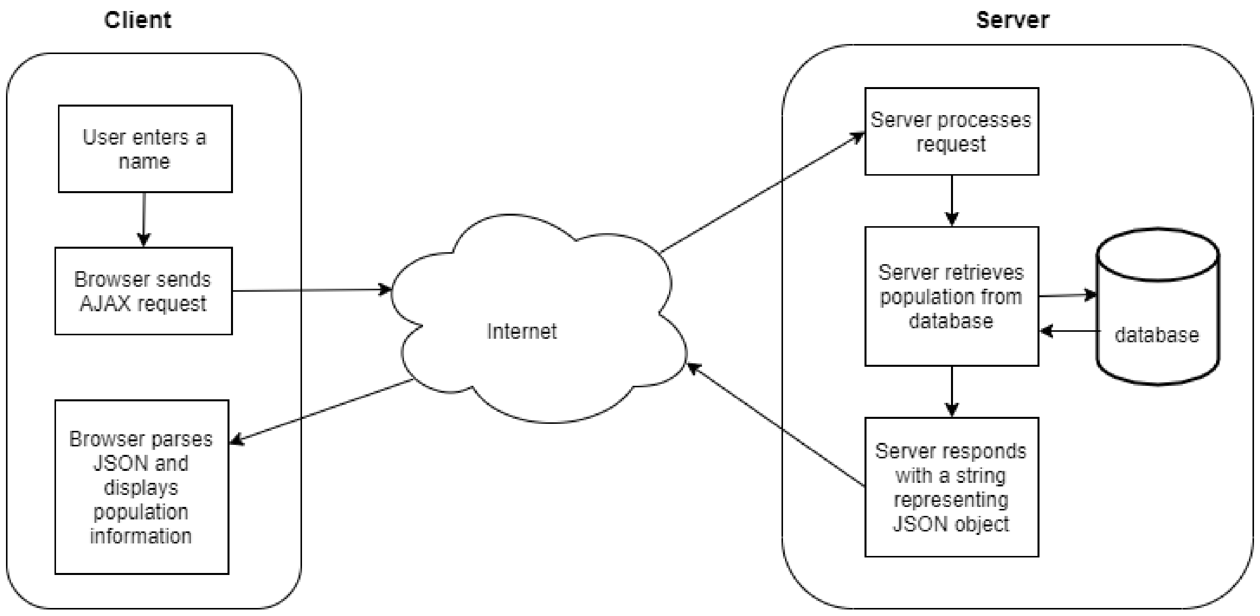
An AJAX Application for Looking up population figures by country/region names

Enter country/region name (e.g. Zambia): Zambi Name not found

Country/Region: **Code:**

Year	Country/Region Population	World Population

Program Flow


Client
User enters a name
Browser sends AJAX request
Browser parses JSON and displays population information
Internet
Server
Server processes request
Server retrieves population from database
database
Server responds with a string representing JSON object

The Data Layer

The application stores its data in a MySQL database named **world_population**. The dataset was downloaded from https://datahub.io/core/population/ as a csv format and then converted into a MySQL database. The database contains one table called **population**.

The Composition

The application is composed of the following files:

- *index.php*: As usual, this file presents the interface of the application.
- *main.js*: This file handles all client-side tasks. Main tasks include sending AJAX requests and receiving server responses and displaying population figures.
- *population.class.php*: This file defines a class named **Population**. Its object is responsible for connecting to the MySQL database server and selecting the **world_population** database. It also provides a public method named **lookup**. The method accepts a name as the parameter and filters the data in the **population** table with the name. It returns an associative array that contains the population information if the name is found or one element with the key named "error" if the name cannot be found.
- *population.php*: This file handles the server-side scripting. It retrieves the name from a query string variable sent by an AJAX request, passes it to the **lookup** method defined in a **Population** object, and outputs a string representation of a JSON object.
- main.js: The JavaScript file for handling client-side scripting.
- main.css: The css file for styles.

The JSON object

The *population.php* sends a JSON string in response to an ajax request. The following is a part of the JSON string for the country Zambia. Please note the last element of the JSON contains metadata of the country/region's name and code.

```
{
   "1960":
      {
          "World":3034193297,
          "Zambia":3044846
      },
    "1961":
      {
          "World":3075115342,
          "Zambia":3140264
      },

    ......

    "2016":
      {
          "World":7442135578,
          "Zambia":16591390
      },
     "metadata":
     {
         "name": "Zambia",
         "code": "ZMB"
     }
}
```

Step-by-Step Instructions

1. Download the data files from Canvas and extract the **Lab05** folder into **htdocs/I211** folder. One of the files included in the download is named *world_population.zip*, which compresses *world_population.sql*.
2. Create the **world_population** database by importing the *world_population.zip* file.
3. Open *population.class.php* file in PhpStorm. Modify the **login** and **password** inside the class constructor to use an account available on your MySQL server.
4. Modify the *index.php* file.
   a. Link the *index.php* file to the external JavaScript file named *main.js*. This should be done in the document's head section.
   b. Locate the <input> tag that creates the textbox whose id is **name**. Modify the tag by adding an event handler so that whenever a keystroke is detected, the JavaScript function **handlekeyup** is invoked. Please note the function accepts one parameter named **event**. You should not rename this parameter.

5. Complete the **handlekeyup** function in *main.js*.

   This function defines and sends a new asynchronous AJAX request to the server and then handles server's responses. The function accepts one parameter called **name**.
   - The server script that accepts the request is named *population.php*, which needs to be completed in the next step. Along with every request, one query string variable named **name** should be sent. The value of the query string variable is the parameter of the function called **name**.
   - To handle server's responses, parse the JSON object returned from the server and then pass the JSON object to the **display** function.
   - Please note when the country/region name cannot be found in the database, the JSON object received from the server should contain a key named "error" and the value is an error message. You need to call the **error** function to display the error message. To determine whether a JSON object has a particular key, you may use the **hasOwnProperty** method. This function returns a Boolean value. For more details, please visit https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty.
6. Create the *population.php* file. Refer to the **Composition** section explained above for tasks this script file needs to carry out. Please review the code in *guess.php* file, which is part of the Guessing the Number game we completed in class.

   After you have completed the above steps, you should test your application by running *index.php*. Make sure you open the **Developer tools** in your browser and then the **Network** tab. Type a country name into the textbox on the webpage and watch the AJAX requests in the **Network** tab. If your code is correct, you should see AJAX requests and server's response in the **Network** tab. If you see nothing or errors, please fix the error. You need to switch to the **Console** tab to view JavaScript errors. Once you have fixed all the errors and made sure server's response is correct, more forward to the next step.

7. Complete the **display** function in *main.js*.

   The function accepts a JSON object containing population information for a country/region and displays the country/region name, code, and population figures in a div block with the id "population-results". Refer to the JSON object explained in the **JSON Object** section above. The following code adds new row:

   ```
   <div class='row'>
       <div>Year</div>
       <div>Country population</div>
       <div>World population</div>
   </div>
   ```

8. Complete the **clear** function in *main.js*. This function clears the name, code, and population figures.

9. Complete the **error** function in *main.js*. This function displays whatever is passed to it in a div with the id of “message”.

Code style guidelines:

1. Code style is as important as the code itself. Code style includes enough comments, adequate space between code blocks, and proper indentation, etc. Read details and view sample code from http://pear.php.net/manual/en/standards.php.
2. Please provide sufficient comments in your source code. Source code is a language for people, not just for computers. Comment as you go and don’t wait for later. Ask yourself: “How will the next person know that?” Commenting code shows your professionalism, but also helps your grader understand your code.
3. Every class file should contain a header in this format:
   /*
   * Author: *your name*
   * Date: *today’s date*
   * Name: *file name*
   * Description: short paragraph that explains what the class is for
   */
4. Indent your code. Leave enough space between code blocks. You can use the **Reformat Code** command (*Code* > *Reformat Code*) in PhpStorm to automatically format your code.

Turning in your lab

Your work will be evaluated on completeness and correctness. Thoroughly test your code before you turn it in. It is your responsibility to ensure you turn in the correct files. You will NOT receive any credit if you turn in the wrong files whether or not you’ve completed the lab.

1. Zip the entire **Lab05** folder and save it as *Lab05.zip*.
2. Upload the *Lab05.zip* file in Canvas before the lab’s deadline.

Grading rubric

Your TAs will assess your lab according to the following grading rubric. You should very closely follow the instructions in this handout when working on the lab. Small deviations may be fine, but you should avoid large deviations. You will not receive credits if your deviation does not satisfy an item of the grading rubric. Whether a deviation is small or large and whether it satisfies the requirement are at your TAs’ discretion. Here is the breakdown of the scoring:

Modifying *index.php* (1 point)

Activities	**Points**
Modify the div block to handle keyup event	1

Modifying *main.js* (11 points)

Activities	**Points**
Create a **XMLHttpRequest** object	1
Define (open) an asynchronous AJAX request	2
Send the AJAX request	1
Handle server's responses	3
Complete the **display** function	2
Complete the **clear** function	1
Complete the **error** function	1

Creating *population.php* (3 points)

Activities	**Points**
Retrieve the query string variable named **name**	1
Filter the data in the **world_population** database table with a name	1
Parse and output a JSON object	1

Programming style (5 points)

Activities	**Points**
Comment your code	3
Use white spaces to separate code sections	1
Indent and line up code	1