

PHP MySQL Ajax Live Search

In this tutorial you'll learn how to create a live MySQL database search feature using PHP and Ajax.

Ajax Live Database Search

You can create a simple live database search functionality utilizing the Ajax and PHP, where the search results will be displayed as you start typing some character in search input box.

In this tutorial we're going to create a live search box that will search the *countries* table and show the results asynchronously. But, first of all we need to create this table.

Step 1: Creating the Database Table

Execute the following SQL query to create the *countries* table in your MySQL database.

Example	Download
<pre>CREATE TABLE countries (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, name VARCHAR(50) NOT NULL);</pre>	


After creating the table, you need to populate it with some data using the [SQL INSERT statement](#). Alternatively, you can download the prepopulated *countries* table by clicking the download button and import it in your MySQL database.

Please check out the tutorial on [SQL CREATE TABLE statement](#) for the detailed information about syntax for creating tables in MySQL database system.

Step 2: Creating the Search Form

Now, let's create a simple web interface that allows user to live search the names of countries available in our *countries* table, just like an autocomplete or typeahead.

Create a PHP file named "search-form.php" and put the following code inside of it.

Example	 Download
<pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>PHP Live MySQL Database Search</title> <style type="text/css"></pre>	

```

body{
    font-family: Arial, sans-serif;
}
/* Formatting search box */
.search-box{
    width: 300px;

```

Every time the content of search input is changed or keyup event occur on search input the jQuery code (line no-47 to 67) sent an Ajax request to the "backend-search.php" file which retrieves the records from *countries* table related to the searched term. Those records later will be inserted inside a `<div>` by the jQuery and displayed on the browser.

Step 3: Processing Search Query in Backend

And here's the source code of our "backend-search.php" file which searches the database based on query string sent by the Ajax request and send the results back to browser.

Example	Procedural	Object Oriented	PDO	Download
<pre> <?php /* Attempt MySQL server connection. Assuming you are running MySQL server with default setting (user 'root' with no password) */ \$link = mysqli_connect("localhost", "root", "", "demo"); // Check connection if(\$link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); } if(isset(\$_REQUEST["term"])){ // Prepare a select statement \$sql = "SELECT * FROM countries WHERE name LIKE ?"; </pre>				

The SQL `SELECT` statement is used in combination with the `LIKE` operator (line no-16) to find the matching records in *countries* database table. We've implemented the `prepared statement` for better search performance as well as to prevent the `SQL injection` attack.

Note: Always filter and validate user input before using it in a SQL statement. You can also use PHP `mysqli_real_escape_string()` function to escape special characters in a user input and create a legal SQL string to protect against SQL injection.