GET – collecting form data

Welcome to this lab activity

In this lab activity, you will explore how to collect data from your existing form using the GET request method.

Task 1: Make a copy of your previous lab

You will start with your previous lab code.

1. Make a copy of the folder `08_forms` from your previous lab and call it `08_get_form_data`.
2. Open up the new folder in Visual Studio Code.
3. Run the code and confirm that you can see the homepage and about page (refer to the previous lab if you need a reminder of how to do this).

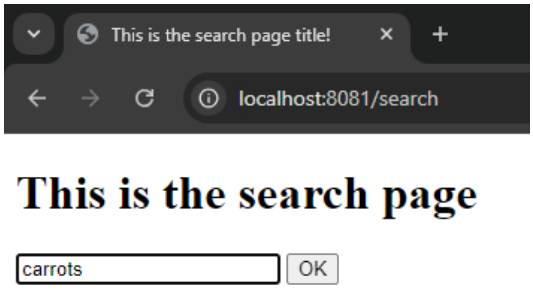
Task 2: Collecting form data using a GET request

Look at the form inside `search.ejs`:

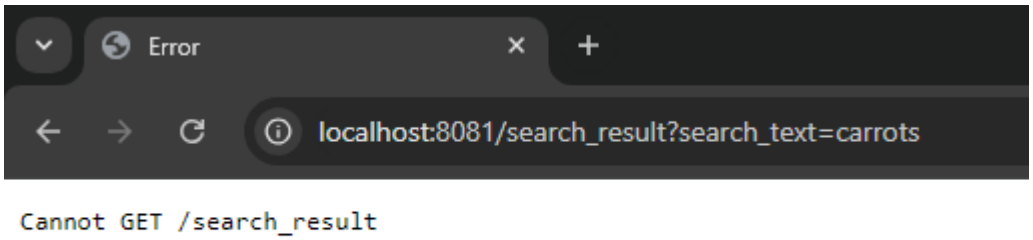
```
<form action="/search_result" method="GET">
    <input id="search_box" type="text" name="search_text">
    <input type="submit" value="OK" >
</form>
```

You will notice that the method is GET. This signifies that when the user submits the form, it will be submitted using the HTTP method called GET. This method puts the data from the form in the URL that is submitted to your web server.

4. Run the code (refer to the previous lab if you need a reminder of how to do this).
5. Go to the search page and enter your search text, for example:



6. Submit the form by pressing the ‘OK’ button. You should see something like this:



You can see above that the URL has changed to `/search_result?search_text=carrots`. The search text from the form has been added to the URL and the route is now `/search_result`, which is the action on the form. The text after the route, `?search_text=carrots`, is called a **query string**.

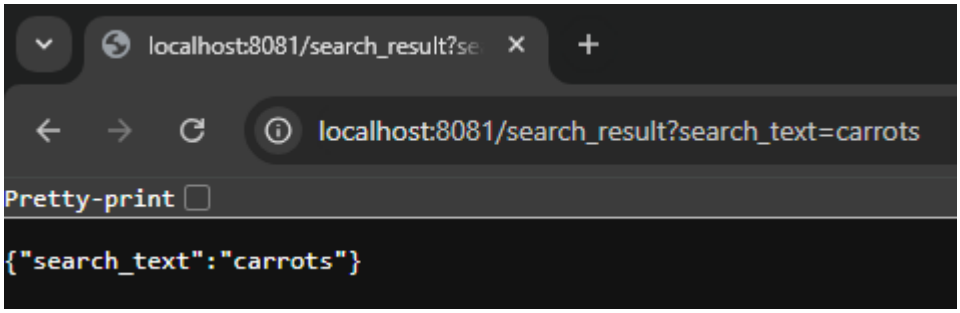
To access the search text, you need a new route handler that can handle the search-result route and pull out the search text from the query string.

7. Add the following code to your `main.js` file:

```
router.get("/search_result", function (req, res) {
    res.send(req.query);
});
```

As you can see, this code handles the `/search_result` route. To do so, it extracts the `query` object from the `req` (request) object and sends it back in the response. Let’s see this code in action.

8. Stop your web server and start it up again so the above code changes take effect.
9. Run the code, go to the search form and add some search text. Then press ‘OK’ to submit the form. You should see something like this:



Here, the browser has displayed the request it received –the `req.query` object – which has a value `{"search_text":"carrots"}`.

Task 3: Extract specific values from the query

If your form has multiple fields, your query string will contain all the fields submitted in the form and the query object will contain these same fields. For example, if your form had another field called 'category', your query string might look like this:

```
?search_text=carrots&category=vegetables
```

And your query object would then look like this:

```
{
  "search_text":"carrots",
  "category":"vegetables"
}
```

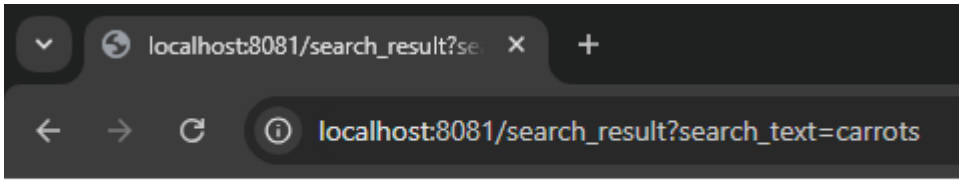
Within your route handler code, you will want to extract the specific fields from the query object. Let's do that now.

10. Change the `/search_result` route handler to pull out the `search_text` field only:

```
router.get("/search_result", (req, res) => {
    res.send("You searched for " + req.query.search_text);
});
```

11. Run the code, go to the search form and enter some search text. Then submit the form.

You should see just the search text displayed:



Task 4: Explore further

When tackling these lab activities, it's always good to stretch yourself by doing some research and attempting some changes on your own.

In Task 3 discussed a scenario where there are multiple form fields. Implement this additional field and output the value of it in your GET route handler.

End of lab

Congratulations on completing this lab.

You have successfully created a web application that submits data in a HTML form. Furthermore, you are now able to collect that data using the GET request method.

In the next lab activity, you will explore another important method for data manipulation: the POST method.