Lab 3

Due Mar 18 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

CS-554 Lab 3

Redis

For this assignment, you will practice your usage of redis and `async/await` together. You will promisify the redis-client for use, and you will use all of its methods with the use of the `await` keyword. You will also use handlebars and `axios` to get our data from the API.

Handlebars

You will need to use Handlebars for this lab. Do NOT output raw JSON data.

Our Data

For this lab, we are going to use an API to get our data. We will be using the **TV Maze API** **(http://api.tvmaze.com)**

We will specifically be using three endpoints:

http://api.tvmaze.com/shows **(http://api.tvmaze.com/shows)** - This URL displays a list of TV Shows

http://api.tvmaze.com/shows/:id - This URL displays the show details for the specified `:id`
http://api.tvmaze.com/shows/1 **(http://api.tvmaze.com/shows/1)** is an example of the show Under the Dome

http://api.tvmaze.com/search/shows?q=SEARCH_TERM - This URL is for searching shows where you pass in the term you are searching for in the query string parameter. **http://api.tvmaze.com/search/shows?q=Under** **(http://api.tvmaze.com/search/shows?q=Under)** is an example of searching the shows for the word Under

Hint: Look at what your data is returning as you will need to understand the structure of the data to be able to parse it and get the data that you need.

Your cache

We will be caching fully rendered HTML pages from handlebars templates in this lab as well as keeping track of the most searched search terms in a list with a count of how many times a search term was searched for. You will need to research the optional callback function that `res.render()` takes and use that to get the RAW html output that was rendered by `res.render()` and then use `res.send()` to send that rendered HTML data as a response to the client in that callback.

For the most searched search terms, you will need to research **sorted sets** in Redis (these are sometimes referred to as scoreboards or leaderboards) We will be storing the search term and then a count of how many times that search

term was searched for. If someone searches for a term that has been searched for already, you will increment the count. The search term will be the key and the count the value. If they search for a term not in the list, you will add it to the list and set the count initially to 1.

Your Routes

You will have 4 routes:

`/`

GET:

This route will show the list of shows. It will check the cache to see if we have the show list homepage already cached. If the show list homepage is already in the cache, you will serve it from the cache.

If the show list homepage is NOT in the cache, it will query the API to list all the shows, it will also render a handlebars template, passing in the list of shows and then store the rendered HTML in cache, and send that raw rendered HTML to the client.

Note: For the handlebars template, you will just display a list with the show name as the list item. That show name will be hyperlinked to /show/:id where `:id` is the ID of the show (the ID is available in the data in the list of shows). You must render a full HTML page, no raw JSON data just dumped to the page.

`/show/:id`

GET:

This route will show the details of a show. It will check the cache to see if we have the show details page already cached. If so, you will serve it from the cache.

If the show details page is NOT in the cache, it will query the API based on the `:id` for that show, it will also render a handlebars template, passing in the show details returned from the API and then store the rendered HTML in cache, and send that raw rendered HTML to the client.

Note: For the handlebars template. Please display as much of the show detail data as possible. Show the TV show name, the summary, show the image and other details about the show that are found in the data for the show. You must render a full HTML page, no raw JSON data just dumped to the page.

If no show can be found for the given ID, then display an error to the user with a status code 404.

`/search`

POST:

This route will search the data based on the searchTerm that is passed into the request.body

You will display a search form on ALL pages of the site(so put it in your main handlebars layout). When the user enters a search term and submits the form, the search term will be posted to this route.

This route will need to make sure that a search term was provided in the req.body, if it is not provided, then display an error to the user (make sure to take into account a string with empty spaces " " should not be valid).

This route will do a few things and require at least two different Redis operations:

It will check to see if the search term the user searched for is in our sorted set. If the term is not in the set, you will add it to the set and set the initial value to 1.

If that search term is in the set then you will increment the search term counter in the sorted set for that search term.

After you have done that you will check the cache to see if it has search results for that search term. If it does you will send those cached results to the client.

If there are no results in the cache for that search term, you will then query the API endpoint for the search term and then render a handlebars template passing in the results of the search query that axios returns. You will then cache the rendered HTML search results for that search term (this way, if someone else searches that same term, we can give them the results from the cache) and then send that raw rendered HTML to the client.

Note: For the handlebars template, you will just display a list with the show name as the list item just like you did for the show list on the homepage. That show name will be hyperlinked to /show/:id where :id is the ID of the show (the ID is available in the data in the list of shows). You must render a full HTML page, no raw JSON data just dumped to the page.

`/popularsearches`

GET:

This route will display the top 10 search terms that are stored in our sorted set.

You will return the 10 search terms from our sorted set with the highest number of searches being first in the list.

You will then pass in the top 10 search terms into a handlebars template and display them in ol or ul

Note: You must render a full HTML page, no raw JSON data just dumped to the page. This page does not have to be cached.

General Requirements

1. Remember to submit your `package.json` file but **not** your `node_modules` folder.

2. Make sure you display an error if a show cannot be found **http://localhost:3000/show/BAD_ID (http://localhost:3000/show/BAD_ID)**
3. Make sure you display an error if there is no search term entered into the form or if they type empty spaces into the form: " "
4. You must make sure ALL your HTML is valid HTML
5. You must make sure all pages pass Tota11y tests.