



# CS-602: Server-Side Web Development

## Assignment 2

This document should not be disseminated outside the purview of its intended purpose.

### General Rules for Homework Assignments

- You are strongly encouraged to add comments throughout the program. Doing so will help your instructor to understand your programming logic and grade you more accurately.
- You must work on your assignments individually. You are not allowed to copy the answers from others.
- Each assignment has a strict deadline. Assignments submitted after the deadline will carry a penalty.
- When the term `lastName` is referenced in an assignment, please replace it with your last name.

Download and extract the starter template zip file, `CS602_HW2_lastName`. Rename the folder with your last name. Complete the corresponding JavaScript and View files in this folder.

### Part 1 – Net Module (25 Points)

Use the subfolder `part1` and complete the following files. Copy the `zipCodeModule_v2.js` from Module1 assignment. Complete the server application (`server/server.js`) and the client application (`client/client.js`) using the net module. The clients communicate with the server using the following commands:

- `lookupByZipCode, <zip>`
- `lookupByCityState, <city>, <state>`
- `getPopulationByState, <state>`



The **server** application listens for client connections and when the client command is received, processes the request by invoking the corresponding function available through the **zipCodeModule**. The server then uses `JSON.stringify` to send the result back to the client.

The **client** application reads commands from the user's console in a loop and sends the commands to the server. When the data is received back from the server, the corresponding result is printed to the console.

Test the application with the server running in one window, and a client running in a separate window.

The sample output of the server and the client application is shown below. You can optionally use the `colors` module for colors in the output.

```
> node server.js
Listening for connections on port 3000
Client connection...
Client connection...
...Received lookupByZipCode,02215
...Received lookupByZipCode,99999
...Received lookupByCityState,BOSTON,MA
...Received lookupByCityState,BOSTON,TX
...Received getPopulationByState,MA
...Received getPopulationByState,TX
...Received lookupByState,MA
```

```
> node client.js
Connected to server
Enter Command: lookupByZipCode,02215
...Received
{"_id":"02215","city":"BOSTON","pop":17769,"state":"MA"}
Enter Command: lookupByCityState,BOSTON,MA
...Received
{"city":"BOSTON","state":"MA","data":[{"zip":"02108","pop":3697},
{"zip":"02109","pop":3926},{"zip":"02110","pop":957},{"zip":"02111",
"pop":3759},{"zip":"02113","pop":6698},{"zip":"02114","pop":1024
6},{"zip":"02115","pop":25597},{"zip":"02116","pop":17459},{"zip":
"02199","pop":886},{"zip":"02210","pop":308},{"zip":"02215","pop":
17769}]}
Enter Command: getPopulationByState,MA
...Received
{"state":"MA","pop":6016425}
Enter Command: lookupByState,MA
...Received
"Invalid request"
Enter Command: █
```

## Part 2 – Express, Handlebars & REST Endpoints (75 Points)

Use the subfolder part2 and complete the following files. Copy the `zipCodeModule_v2.js` from Module1 assignment. Complete the Express server application (`server.js`) and the corresponding views to do the following:

- GET request – /

Render the home view with a welcome message.

- GET request – /zip  
If the request query **id** parameter is present, lookup the corresponding data and render the **lookupByZipView**. Otherwise, render the **lookupByZipForm**.
- POST request – /zip  
Lookup the corresponding data for the request body **id** parameter and render the **lookupByZipView**.
- GET request – /zip/:id  
Should be capable of handling JSON, XML, and HTML requests. Use the named routing **id** parameter and lookup the corresponding data. For HTML request, render the **lookupByZipView**.
- GET request – /city  
If the request query **city** parameter and **state** parameter are present, lookup the corresponding data and render the **lookupByCityStateView**. Otherwise, render the **lookupByCityStateForm**.
- POST request – /city  
Lookup the corresponding data for the request body **state** and **city** parameters and render the **lookupByCityStateView**.
- GET request – /city/:city/state/:state  
Should be capable of handling JSON, XML, and HTML requests. Use the named routing **city** and **state** parameters and lookup the corresponding data. For HTML request, render the **lookupByCityStateView**.
- GET request – /pop  
If the request query **state** parameter is present, lookup the corresponding data and render the **populationView**. Otherwise, render the **PopulationForm**.
- GET request – /pop/:state  
Should be capable of handling JSON, XML, and HTML requests. Use the named routing **state** parameter and lookup the corresponding data. For HTML request, render the **populationView**.

## Testing the Application (via Browser)

Test the application for HTML requests using the browser. The default home page can be as shown below.

← → ↻ 🏠 localhost:3000



## CS602 HW2 - Your Name

[Lookup By ZipCode](#) | [Lookup By City,State](#) | [Get State Population](#)

### Home Page for CS602 Assignment2

Click the *Lookup By ZipCode* link. The following form is displayed to the user.

← → ↻ 🏠 localhost:3000/zip



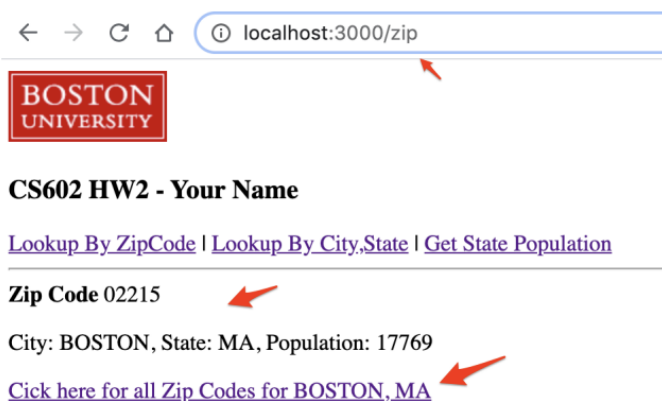
## CS602 HW2 - Your Name

[Lookup By ZipCode](#) | [Lookup By City,State](#) | [Get State Population](#)

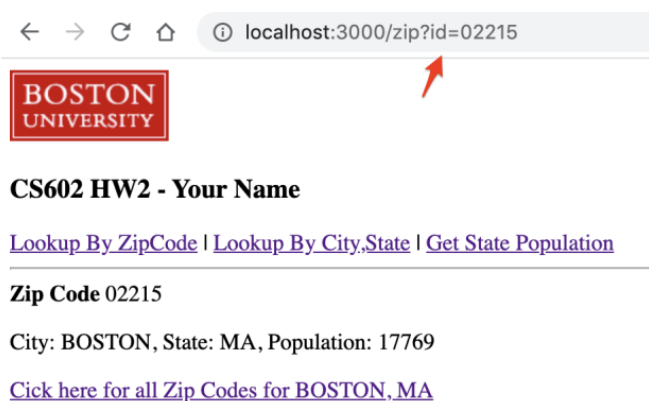
### Lookup By Zip Code

ZipCode:

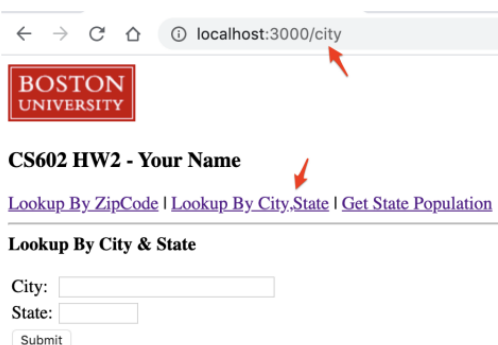
If the user enters, say 02215 for ZipCode and clicks the **Submit** button, the above form is submitted with the POST request. The resulting view is as shown below:



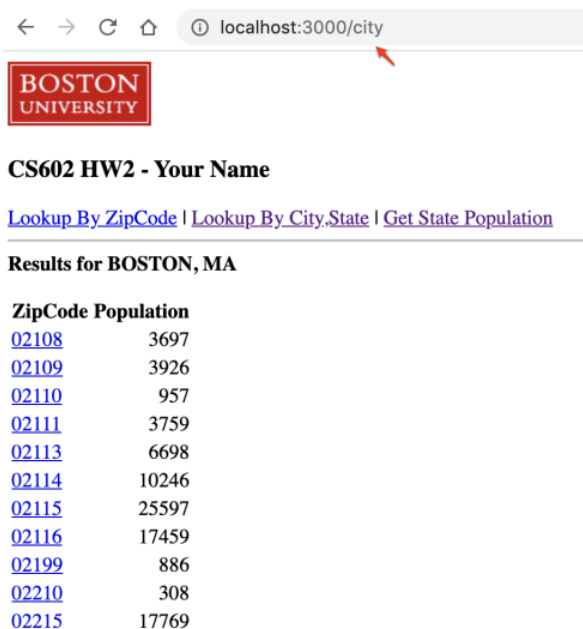
If the user types the direct URL as shown below, the resulting view is as below:



Click the *Lookup By City,State* link. The following form is displayed to the user.

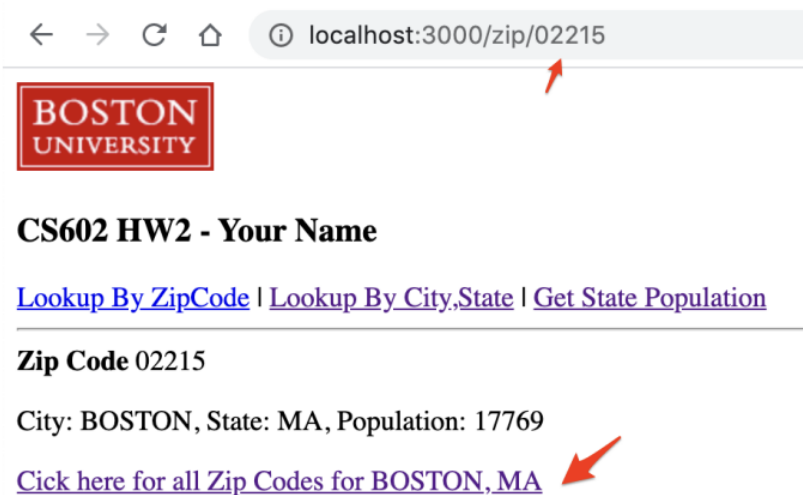


If the user enters, say BOSTON for **City**, MA for **State**, and clicks the **Submit** button, the above form is submitted with the POST request. The resulting view is as shown below:



ZipCode	Population
<a href="#">02108</a>	3697
<a href="#">02109</a>	3926
<a href="#">02110</a>	957
<a href="#">02111</a>	3759
<a href="#">02113</a>	6698
<a href="#">02114</a>	10246
<a href="#">02115</a>	25597
<a href="#">02116</a>	17459
<a href="#">02199</a>	886
<a href="#">02210</a>	308
<a href="#">02215</a>	17769

If the user clicks on a particular zip code in the above view, the result is processed with a GET request as shown below.

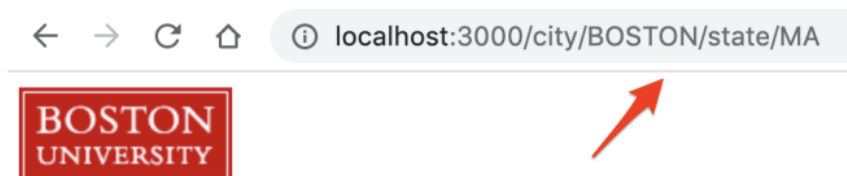


**Zip Code 02215**

City: BOSTON, State: MA, Population: 17769

[Click here for all Zip Codes for BOSTON, MA](#)

In the above view, if the user clicks the last link, the result is processed with a GET request as shown below.



## CS602 HW2 - Your Name

[Lookup By ZipCode](#) | [Lookup By City,State](#) | [Get State Population](#)

### Results for BOSTON, MA

#### ZipCode Population

<a href="#">02108</a>	3697
<a href="#">02109</a>	3926
<a href="#">02110</a>	957
<a href="#">02111</a>	3759
<a href="#">02113</a>	6698
<a href="#">02114</a>	10246
<a href="#">02115</a>	25597
<a href="#">02116</a>	17459
<a href="#">02199</a>	886
<a href="#">02210</a>	308
<a href="#">02215</a>	17769

If the user types the direct URL as shown below, the resulting view is as below:



← → ↻ 🏠 localhost:3000/city?city=BOSTON&state=MA



## CS602 HW2 - Your Name

[Lookup By ZipCode](#) | [Lookup By City,State](#) | [Get State Population](#)

### Results for BOSTON, MA

#### ZipCode Population

<a href="#">02108</a>	3697
<a href="#">02109</a>	3926
<a href="#">02110</a>	957
<a href="#">02111</a>	3759
<a href="#">02113</a>	6698
<a href="#">02114</a>	10246
<a href="#">02115</a>	25597
<a href="#">02116</a>	17459
<a href="#">02199</a>	886
<a href="#">02210</a>	308
<a href="#">02215</a>	17769

Click the [Get State Population](#) link. The following page is displayed to the user.

← → ↻ 🏠 localhost:3000/pop



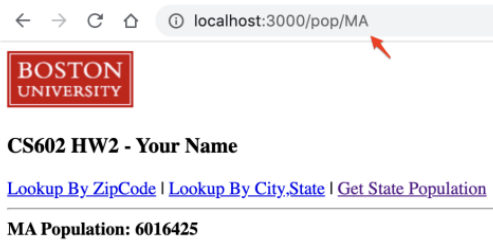
## CS602 HW2 - Your Name

[Lookup By ZipCode](#) | [Lookup By City,State](#) | [Get State Population](#)

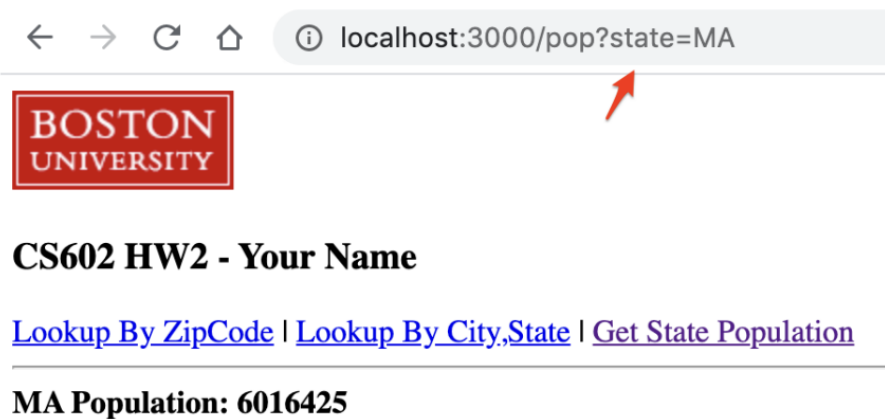
### Click a State

[AK](#) | [AL](#) | [AR](#) | [AZ](#) | [CA](#) | [CO](#) | [CT](#) | [DC](#) | [DE](#) | [FL](#) |  
[GA](#) | [HI](#) | [IA](#) | [ID](#) | [IL](#) | [IN](#) | [KS](#) | [KY](#) | [LA](#) |  
[MA](#) | [MD](#) | [ME](#) | [MI](#) | [MN](#) | [MO](#) | [MS](#) | [MT](#) |  
[NC](#) | [ND](#) | [NE](#) | [NH](#) | [NJ](#) | [NM](#) | [NV](#) | [NY](#) |  
[OH](#) | [OK](#) | [OR](#) | [PA](#) | [RI](#) | [SC](#) | [SD](#) | [TN](#) | [TX](#) |  
[UT](#) | [VA](#) | [WA](#) | [WI](#) | [WV](#) | [WY](#)

If the user clicks on a state, say **MA**, in the above view, the result is processed with a GET request as shown below:



If the user types the direct URL as shown below, the resulting view is as below:



## Testing the Application (REST Endpoints)

The REST endpoints for JSON and XML data can be tested with curl or Postman.

The curl outputs are shown below (If curl is not installed, see <https://curl.haxx.se/download.html>)

Test the application for the JSON GET requests as shown below.

```
> curl -H "Accept:application/json" http://localhost:3000/zip/02215;echo
{"_id":"02215","city":"BOSTON","pop":17769,"state":"MA"}

[> curl -H "Accept:application/json" http://localhost:3000/pop/MA;echo
{"state":"MA","pop":6016425}

[> curl -H "Accept:application/json" http://localhost:3000/city/BOSTON/state/MA;echo
{"city":"BOSTON","state":"MA","data":[{"zip":"02108","pop":3697}, {"zip":"02109","pop":3926}, {"zip":"02110","pop":957}, {"zip":"02111","pop":3759}, {"zip":"02113","pop":6698}, {"zip":"02114","pop":10246}, {"zip":"02115","pop":25597}, {"zip":"02116","pop":17459}, {"zip":"02199","pop":886}, {"zip":"02210","pop":308}, {"zip":"02215","pop":17769}]}
```

Similarly, test the application for the XML GET requests as shown below.

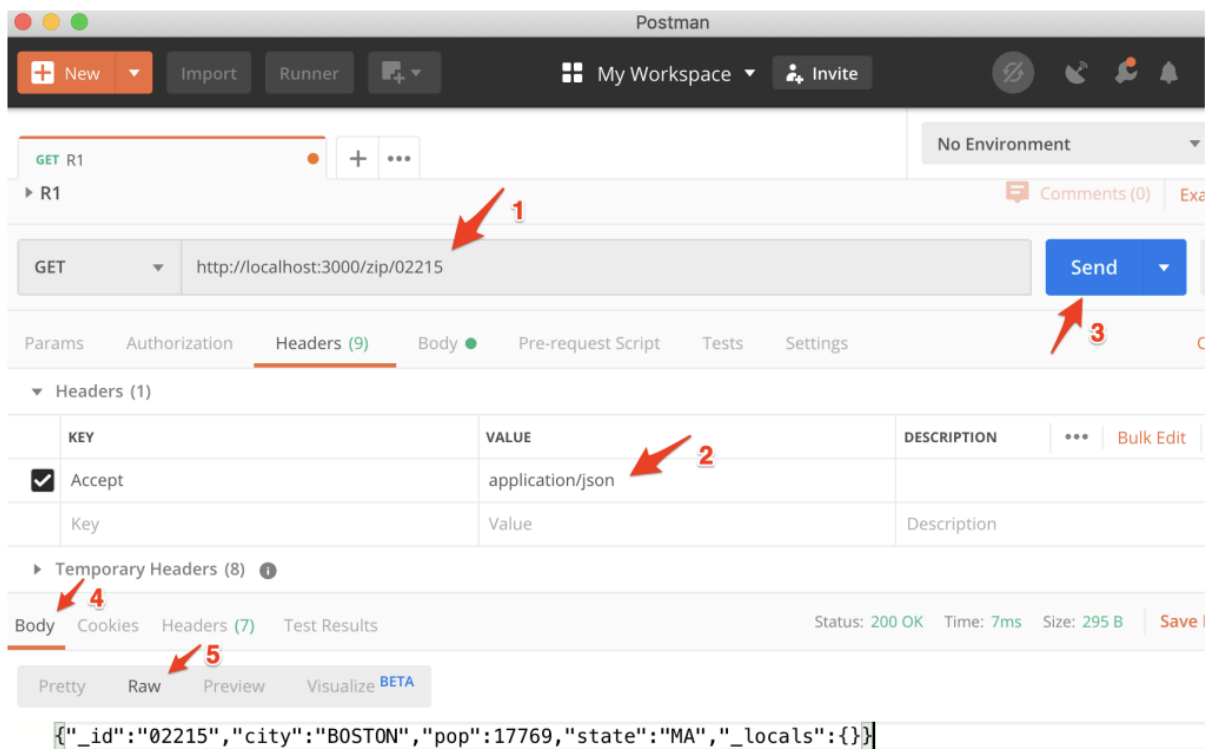
```
[> curl -H "Accept:application/xml" http://localhost:3000/zip/02215
<?xml version="1.0"?>
  <zipCode id="02215">
    <city>BOSTON</city>
    <state>MA</state>
    <pop>17769</pop>
  </zipCode>

[> curl -H "Accept:application/xml" http://localhost:3000/pop/MA
<?xml version="1.0"?>
<state-pop state="MA">
  <pop>6016425</pop>
</state-pop>
```

```
> curl -H "Accept:application/xml" http://localhost:3000/city/BOSTON/state/MA
<?xml version="1.0"?>
<city-state city="BOSTON" state="MA">
  <entry zip="02108" pop="3697" />
  <entry zip="02109" pop="3926" />
  <entry zip="02110" pop="957" />
  <entry zip="02111" pop="3759" />
  <entry zip="02113" pop="6698" />
  <entry zip="02114" pop="10246" />
  <entry zip="02115" pop="25597" />
  <entry zip="02116" pop="17459" />
  <entry zip="02199" pop="886" />
  <entry zip="02210" pop="308" />
  <entry zip="02215" pop="17769" />
</city-state>
```

The Postman outputs for JSON GET requests are shown below.

(For installation of Postman: <https://www.postman.com/downloads/>)



The screenshot shows the Postman application window. At the top, there's a toolbar with 'New', 'Import', 'Runner', and 'My Workspace' buttons. Below this, the 'GET R1' request is selected. The URL bar shows 'http://localhost:3000/zip/02215'. The 'Headers' tab is active, showing a table with one header: 'Accept' with value 'application/json'. The 'Body' tab is also visible, showing the response body in 'Raw' format: `{"_id":"02215","city":"BOSTON","pop":17769,"state":"MA","_locals":{}}`. Red arrows and numbers 1 through 5 highlight specific elements: 1 points to the URL bar, 2 points to the 'Accept' header value, 3 points to the 'Send' button, 4 points to the 'Body' tab, and 5 points to the 'Raw' format button.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/json	
Key	Value	Description

Temporary Headers (8)

Body Cookies Headers (7) Test Results

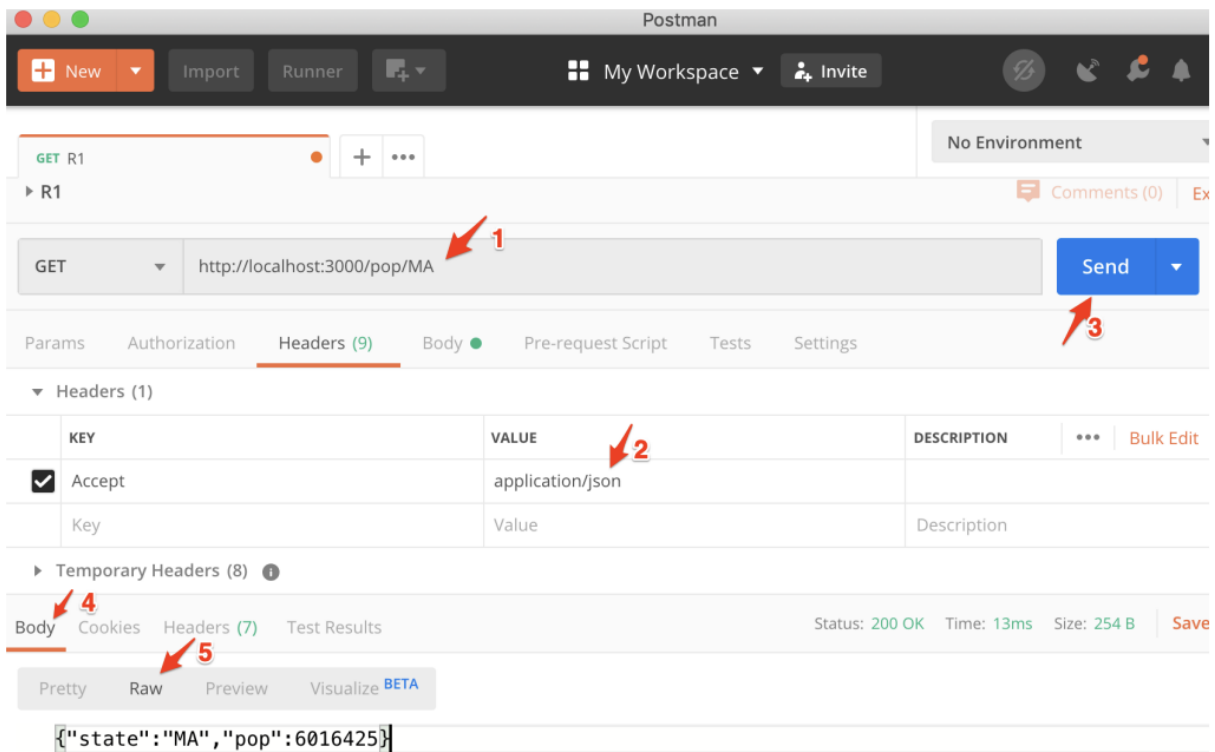
Status: 200 OK Time: 7ms Size: 295 B Save

Pretty Raw Preview Visualize BETA

```

{"_id":"02215","city":"BOSTON","pop":17769,"state":"MA","_locals":{}}

```



The screenshot shows the Postman application window. At the top, there's a toolbar with 'New', 'Import', 'Runner', 'My Workspace', and 'Invite'. Below this, a request is configured as a GET request to the URL `http://localhost:3000/pop/MA`. The 'Headers' tab is selected, showing a table with one header: 'Accept' with the value 'application/json'. The 'Body' tab is also visible, showing the response in 'Raw' format: `{"state": "MA", "pop": 6016425}`. Red arrows and numbers 1 through 5 highlight specific elements: 1 points to the URL, 2 points to the 'Accept' header value, 3 points to the 'Send' button, 4 points to the 'Body' tab, and 5 points to the 'Raw' sub-tab.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/json	
Key	Value	Description

Temporary Headers (8)

Body Cookies Headers (7) Test Results

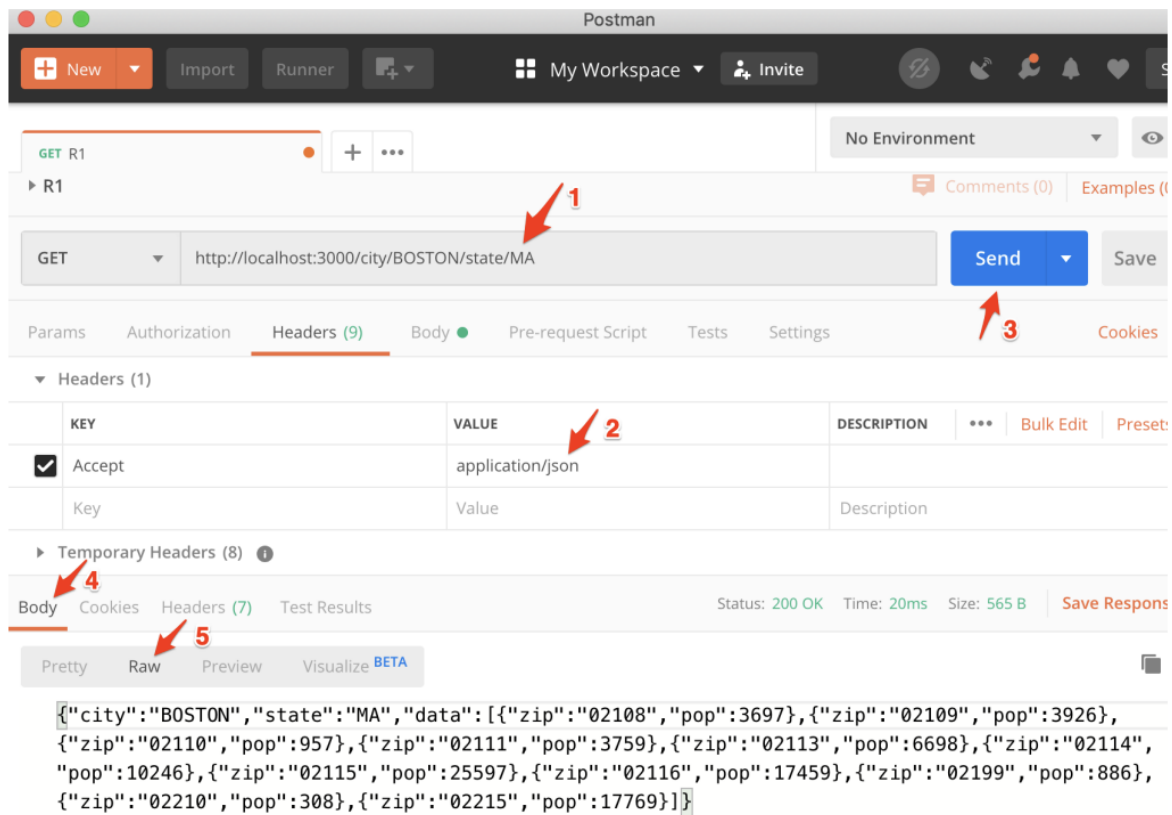
Status: 200 OK Time: 13ms Size: 254 B Save

Pretty Raw Preview Visualize BETA

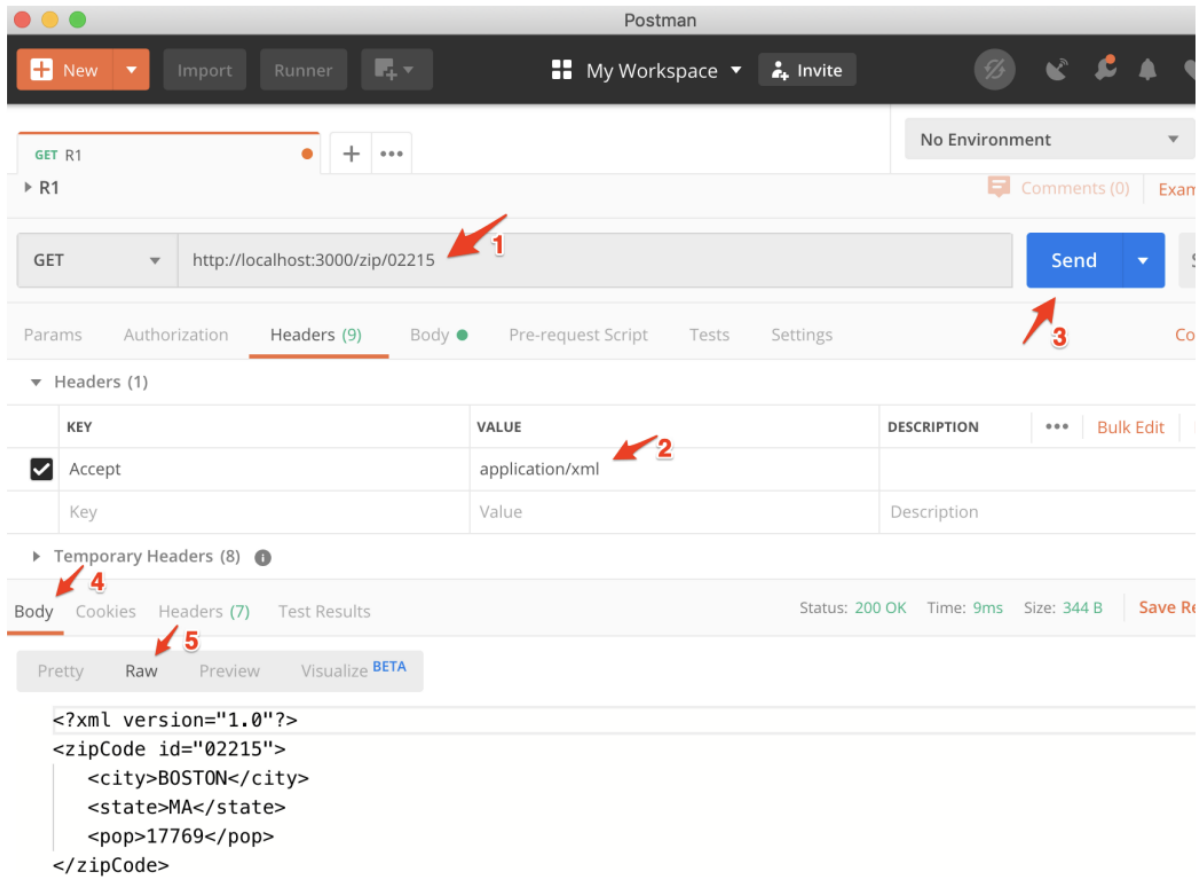
```

{"state": "MA", "pop": 6016425}

```



The Postman outputs for XML GET requests are shown below.

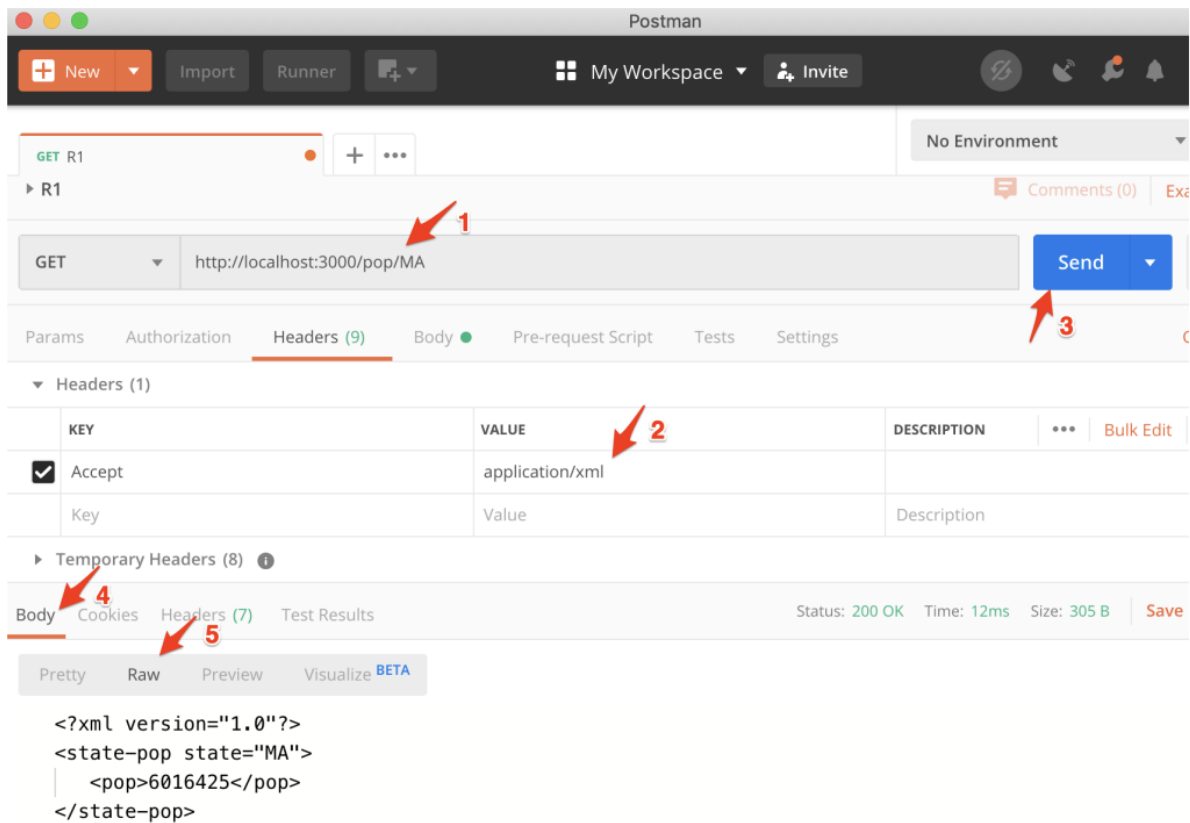


The screenshot shows the Postman application window. At the top, there's a toolbar with 'New', 'Import', 'Runner', and 'My Workspace' buttons. Below this, the 'GET R1' request is selected. The URL bar shows 'http://localhost:3000/zip/02215'. The 'Headers' tab is active, showing a table with headers. The 'Body' tab is also visible, showing the XML response. Red arrows and numbers 1 through 5 highlight specific elements: 1 points to the URL, 2 points to the 'Accept' header value, 3 points to the 'Send' button, 4 points to the 'Body' tab, and 5 points to the 'Raw' sub-tab.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/xml	
Key	Value	Description

```
<?xml version="1.0"?>
<zipCode id="02215">
  <city>BOSTON</city>
  <state>MA</state>
  <pop>17769</pop>
</zipCode>
```





1 points to the URL bar containing `http://localhost:3000/pop/MA`.

2 points to the 'VALUE' column in the Headers table.

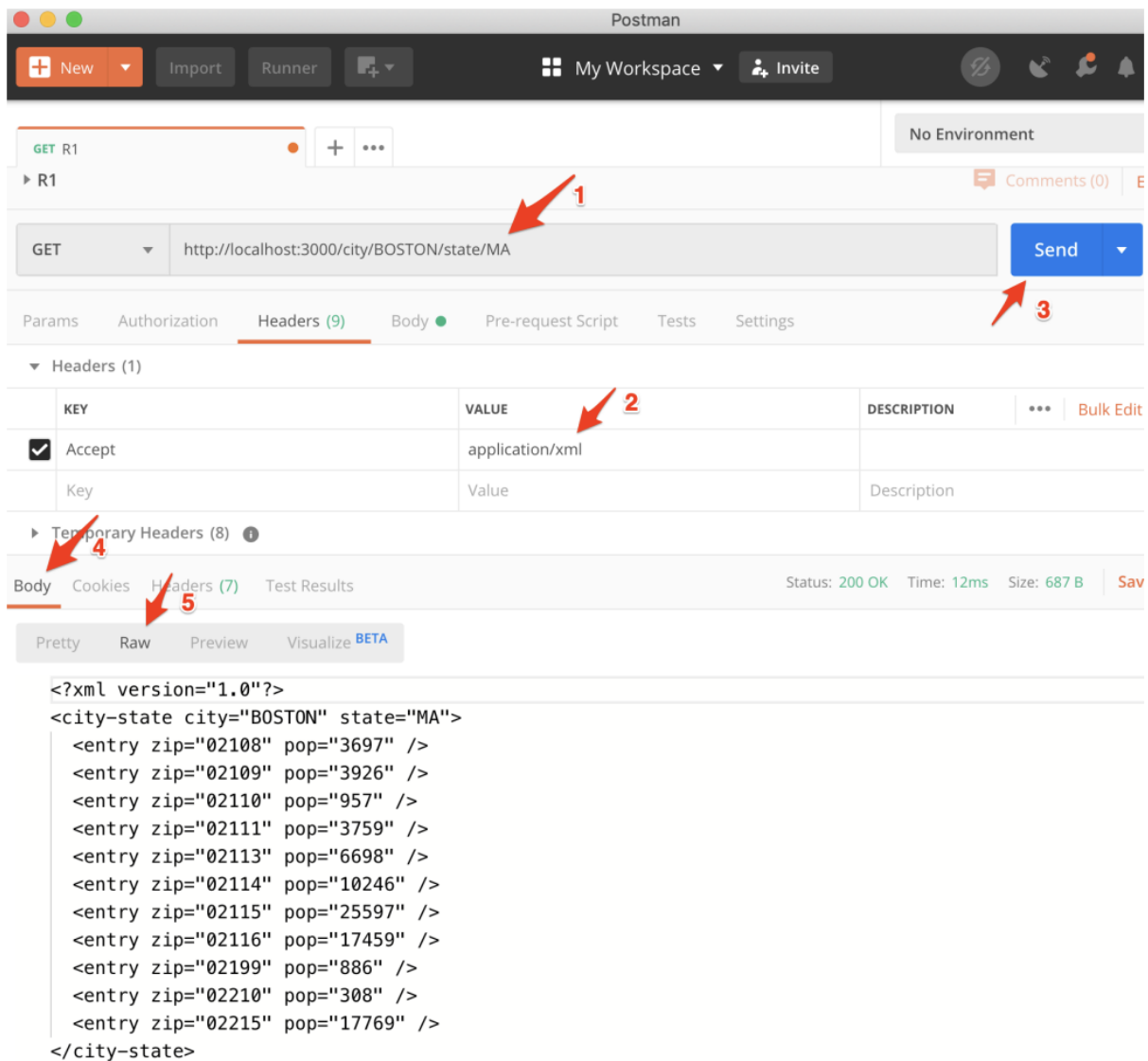
3 points to the 'Send' button.

4 points to the 'Body' tab in the bottom left.

5 points to the 'Headers (7)' tab in the bottom left.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/xml	
Key	Value	Description

```
<?xml version="1.0"?>
<state-pop state="MA">
  <pop>6016425</pop>
</state-pop>
```



## Submission

Export your `CS602_HW2_1stName` folder containing all the relevant files as a zip file, and upload the zip file to the Assignment section.