CS602 Module2 Assignment

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 the others.
- Each assignment has a strict deadline. Assignments submitted after the deadline will have a penalty.
- When the term *lastName* is referenced in an assignment, please replace it with your last name.

Create a new folder/directory named CS602_HW2_*lastName.* Write the following programs in this folder.

Part 1 – net Module (30 Points)

Create a subfolder part1 and write the following files. Install the Node modules `underscore` and `colors` as required. Copy the `employeeModule.js` from Module1 assignment. Using the `employeeModule.js`, write the server application (`server.js`) and the client application (`client.js`) using the `net` module. The clients communicate with the server using the following commands:

- `lookupByLastName` <last name>
- `addEmployee` <first name> <last name>
- `lookupById` <id>

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 `employeeModule`. The server then uses `JSON.stringify` to send the result back to the client.

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

The first client invocation should test the following by reading the user's commands:

- `lookupByLastName Smith`
- `addEmployee William Smith`
- `lookupByLastName Smith`

The first client is then disconnected from the server.

The second client invocation should test the following by reading the user's commands:

- `lookupById 4`

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

```
>node server.js
Listening for connections
Client connection...
...Received lookupByLastName Smith
...Received addEmployee William Smith
...Received lookupByLastName Smith
...Received bye
Client disconnected...
Client connection...
...Received lookupById 4
...Received bye
Client disconnected...
```

```
>node client.js
Connected to server
Enter Command: lookupByLastName Smith
...Received
 [{"id":1,"firstName":"John","lastName":"Smith"}
,{"id":2,"firstName":"Jane","lastName":"Smith"}]
Enter Command: addEmployee William Smith
...Received
 4
Enter Command: lookupByLastName Smith
...Received
 [{"id":1,"firstName":"John","lastName":"Smith"}
,{"id":2,"firstName":"Jane","lastName":"Smith"},
{"id":4,"firstName":"William","lastName":"Smith"
}]
Enter Command: bye
...Received
 "Invalid request"
Client disconnected...


>node client.js
Connected to server
Enter Command: lookupById 4
...Received
 {"id":4,"firstName":"William","lastName":"Smith"}
Enter Command: bye
...Received
 "Invalid request"
Client disconnected...
```

Part 2 – Express, Handlebars & REST (70 Points)

Create a subfolder part2 and write the following files. Install the Node modules `express, express-handlebars, body-parser, and underscore` as required. Copy the `employeeModule.js` from Module1 assignment. Using the `employeeModule.js`, write the Express server application (`server.js`) to do the following:

- GET request – /
  Render the `home` view with a welcome message.
- GET request – /id/:id
  Should be capable of handling json, xml, and html requests.
  Use the `employeeModule` to lookup the employee for the request parameter.
  For html request, render the `employee` view passing the `id` and the `employee` as the model.
- GET request – /lastName/:name
  Should be capable of handling json, xml, and html requests.
  Use the `employeeModule` to lookup the employees for the request parameter.
  For html request, render the `employeeList` view passing the `name` and the `employees` as the model
- GET request – /addEmployee
  Render the `newEmployee` view.
- POST request – /addEmployee
  Use the `employeeModule` to add the employee data from the request body.
  Redirect the user to /lastName/<last name provided by user>

For the views, use the `handlebars` view engine using the default layout `main.handlebars`. The default layout can include your favorite image, a heading with your name, and a link for

/addEmployee. The employeeList.handlebars iterates over the employees and displays the information for each employee. Styling is optional. The employee.handlebars shows the information for the selected employee. The newEmployee.handlebars shows the form for submitting the first name and last name for the new employee. The POST action for this form should be /addEmployee.

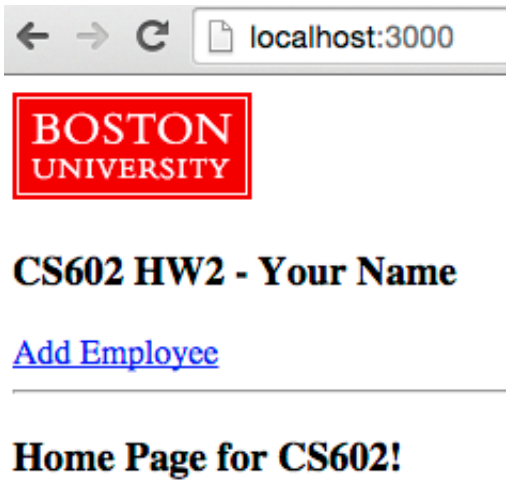
Now, test the application for the json GET requests as shown below.

```
>curl -H "Accept:application/json" http://localhost:3000/lastName/Smith
[{"id":1,"firstName":"John","lastName":"Smith"},{"id":2,"firstName":"Ja
ne","lastName":"Smith"}]>
>
>curl -H "Accept:application/json" http://localhost:3000/id/2
{"id":2,"firstName":"Jane","lastName":"Smith"}>
```

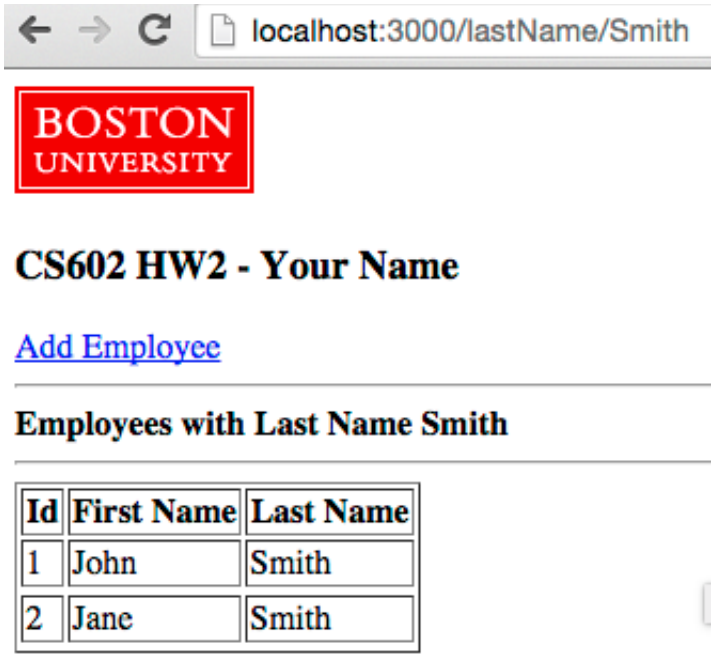
Similarly, test the application for the xml GET requests as shown below.

```
>curl -H "Accept:application/xml" http://localhost:3000/lastName/Smith
<?xml version="1.0"?>
<employees>
 <employee id="1">
   <firstName>John</firstName>
   <lastName>Smith</lastName>
 </employee>
 <employee id="2">
   <firstName>Jane</firstName>
   <lastName>Smith</lastName>
 </employee>
</employees>
>
>curl -H "Accept:application/xml" http://localhost:3000/id/2
<?xml version="1.0"?>
 <employee id="2">
   <firstName>Jane</firstName>
   <lastName>Smith</lastName>
</employee>
```

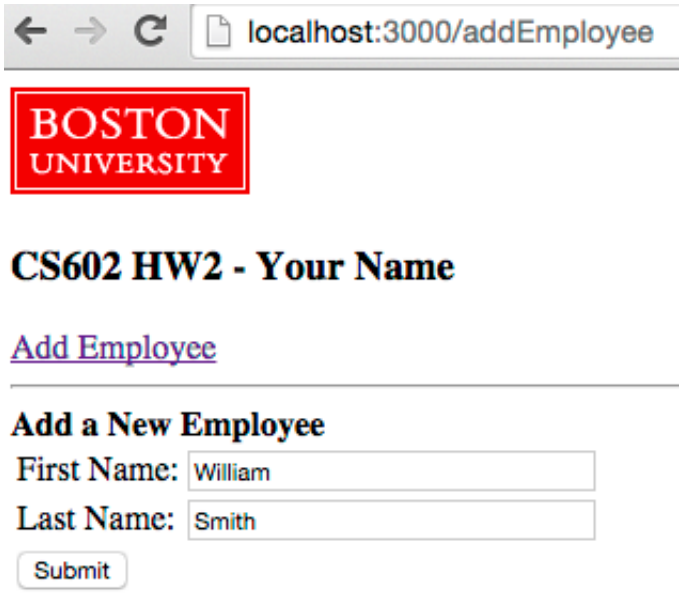
Test the application for html requests using the browser. The default home page can be as shown below.



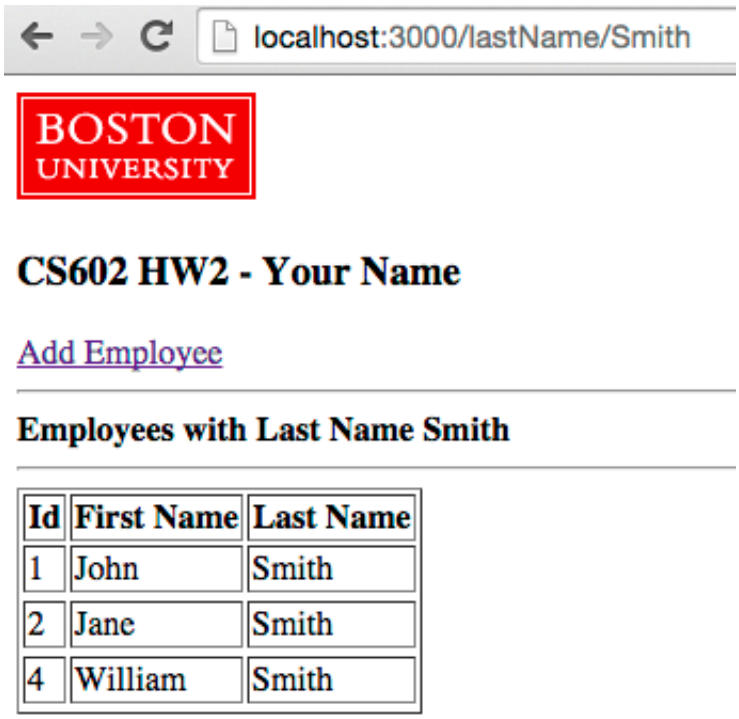
The request for /lastName/Smith is rendered as shown below.



Click the Add Employee link. The form is displayed to the user.



When the above form is submitted, the user is redirected to the corresponding lastName URL.



Submission: Export your CS602_HW2_*lastName* folder containing all the relevant files as a zip file, and upload the zip file to the Assignment section.