RESTFul APIs with Vue

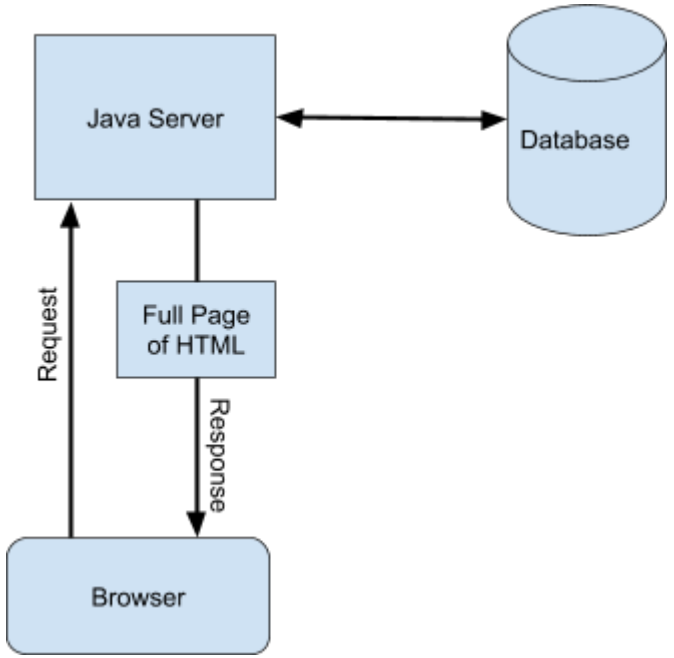
Module 3: 16

Objectives

1. Server Side vs Client Side Applications
2. Axios JS Library
3. Asynchronous vs Synchronous Programming
4. Promises
5. Service Objects
6. GET Requests
7. POST Requests
8. PUT Requests
9. DELETE Requests
10. Error Handling with catch()

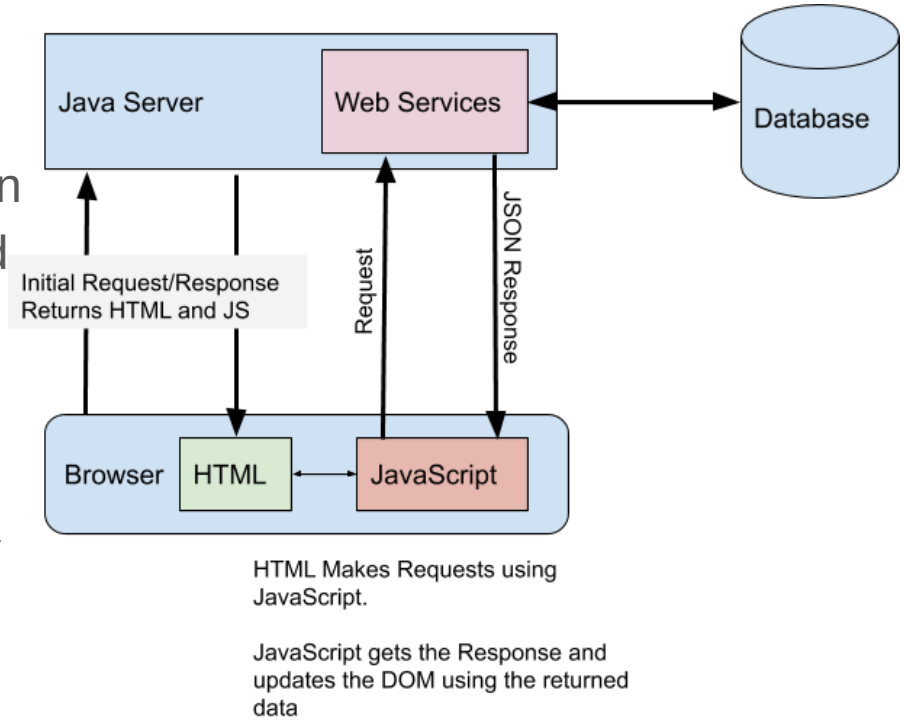
Server Side Application Architecture

1. Route is requested
2. Controller handles the request
3. HTML Response returned
4. Browser gathers resources identified in the HTML (JS, images, CSS, etc.) and renders page to display.
5. When the user needs more data, the whole process repeats.



Client Side with API Application Architecture

1. Route is requested
2. Static HTML page returned
3. Browser gathers resources identified in the HTML (JS, CSS, images, etc.) and renders page for display.
4. When the user more data
    a. JavaScript requests data using an API
    b. Data returned as JSON
    c. JavaScript manipulates the DOM to display changes



Advantages to Client Side Architecture

1. Full page only needs to be served and rendered 1 time
2. Further request/responses to the server will only contain the data
3. Allows for a more feature rich, responsive, and desktop like experience for the user

Workflow

1. HTML page raise Events that will be captured and responded in JavaScript
2. JavaScript will call a Web API with the appropriate request
3. JavaScript will inform us when there is **JSON** response using a **Promise**
4. Once the **JSON** has been returned in the **Promise**, we will use JavaScript to update the DOM to display the new information

Axios JS Library

A Promise based HTTP Client for the browser and Node.js.

Allows for HTTP GET, POST, PUT, and DELETE requests to be made from JavaScript, similar to what the RestTemplate allows for Java.

Automatically converts the response from JSON to JavaScript objects.

Is Asynchronous and returns a Promise to handle the response.

Can be added to a project using: npm install axios

Axios Cheatsheet

Asynchronous vs Synchronous

1. Synchronous Programming
    a. Make a request and wait on the response
    b. While we are response all further execution is held up
    c. This is how method calls in Java and JavaScript work
2. Asynchronous Programming
    a. Make a request and DO NOT wait on the response, instead the ask the responder to tell us when the response ready and then we pause and handle it
    b. Example: Do a fetch, which return a promise that will notify us when the response is ready, while waiting the rest of our code will continue to execute

Promises

- A JavaScript Object that notifies us when an asynchronous process is complete
- Promises have 3 states
  - Pending - initial state, running and neither fulfilled or reject
  - Fulfilled - the operation completed successfully
  - Rejected - the operation failed

Service Objects

A service object in JavaScript is JS file that encapsulates related functionality, for example, an API.

An API Service Object may include the import of the Axios library, setting the BaseURL, creating the Axios object, and exposing methods to be used for the CRUD operations of the API.

GET with Axios

```
axios.get(url)
  .then((response) => {
    console.log(response);
  })
```

url - The full URL of the API

response - The APIs JSON response, converted to a JavaScript object

The **get()** method returns a Promise.

When the Promise is *FullFilled* the method provided to the **.then()** is called.