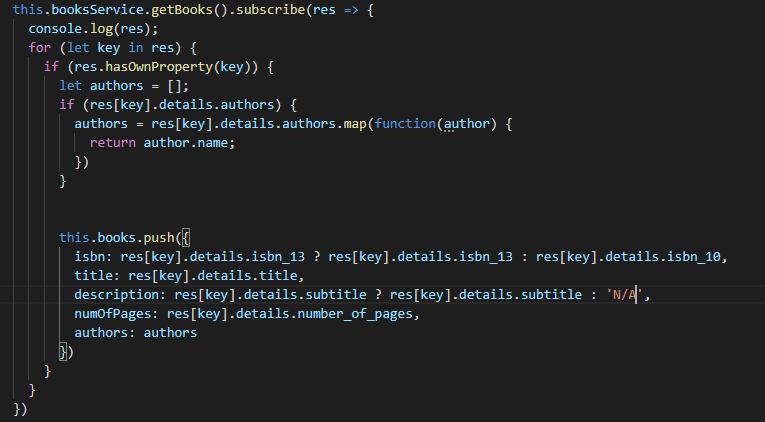
# Instructions – Assignment 8.2 – Server-side Communications

in-n-out-books, final version

**Instructions**

* Make a copy of the in-n-out-booksp4 from Assignment 6.2 and add it to your week-8 directory
* Rename the application to in-n-out-books
* Delete the node\_modules directory
* Delete the package-lock.json file
* Open the angular.json file and find and replace all “in-n-out-booksp4” entries with “in-n-out-books”
* Open the package.json file and change the name to “in-n-out-books”
* Run npm install and ng serve
  + You are doing this to test the application and confirm there are no errors
* Note: if you remember from week six, our in-n-out-books application is using the books.service.ts array of books to populate the book-list.component.html table.  The first part of this exercise is to replace the in-memory data with an API call to the Open Library API
  + <https://openlibrary.org/developers/api>
* app.module.ts
  + Import the HttpClientModule and MatProgressSpinnerModule
    - import { HttpClientModule } from ‘@angular/common/http’;
    - import { MatProgressSpinnerModule } from ‘@angular/material/progress-spinner’;
  + Add the modules to the imports array
* app.component.ts
  + Change the assignment variable’s value to ‘Welcome to In-N-Out-Books”
* books.service.ts
  + Add an import statement for the HttpClient and HttpParams
    - Import { HttpClient, HttpParams } from ‘@angular/common/http’;
  + Remove the books array of type IBook
  + In the services constructor add the HttpClient
    - constructor(private http: HttpClient)
    - Remove the code to populate the array of books
  + Add a new variable named isbns of type string array and fill it with the following values
    - 0345339681,0261103571,9780593099322,9780261102361,9780261102378,9780590302715,9780316769532,9780743273565,9780590405959
  + getBooks()
    - Remove the Observable<IBook[]> return type
    - Create a new variable named params of type HttpParams
      * let params = new HttpParams()
    - Using the params.append function add entries for
      * bibkeys, isbns (use the join function to convert the array of isbns to a comma delimited string)
        + params = params.append(‘bibkeys’,`ISBN:${this.isbns.join(‘,’}`);
      * format, json
        + params = params.append(‘format’, ‘json’)
      * jscmd, details
        + params = params.append(‘jscmd’, details’)
    - return the http.get function passing in the openlibrary url and the params object
      * return this.http.get(‘<https://openlibrary.org/api/books>’, {params: params}
  + Delete the getBook function
* book-list.component.ts
  + Convert the books variable to an array of IBook
  + Remove the code from the constructor’s body
  + Remove the header variable (note: we are rewriting the UI)
  + In the constructors body subscribe the getBooks() function
    - this.bookService.getBooks().subscribe(res => {});
  + In the body of the subscribe function loop over the response data object using a for...in loop and check if the res.hasOwnProperty(key) is true
    - for (let key in res) { if (res.hasOwnProperty(key)) { }}
    - In the body of the if statement push a new object to the books array
      * this.books.push({});



* + - * Note: to access the field values of the object you will need to reference them by dot notation.  If you are not sure of the response output from the API use console.log to see its format
        + console..log(res[key].details);
        + Example: res[key].details.title || res[key].details.isbn\_13[0]
      * Note: this is going to be challenging, so make sure you take your time and try to really understand the format that is being returned from the API.  And, as always, if you have questions post them on Slack
  + showBookDetails() function
    - Replace the code that calls the getBook(isbn) function with a find function
      * this.books = this.books.find(book => book.isbn === isbn);
      * Note: we are simply using JavaScripts built-in find function to search the books array and return the matching book object
* book-list.component.html
  + Delete all the HTML code (Mat table code)
  + Add an <ng-container> give it an ngIf statement to check if the books.length === 0.  In the body of the <ng-container> add a <mat-spinner>
    - <ng-container \*ngIf=”books.length === 0”><mat-spinner></mat-spinner></ng-container>
  + Add a new div and inside the div
    - Add an h2 tag with a color green and name it “Top {books.length} Favorite Books”
      * Note we are using the {books.length} to set the number of books we have in our favorites list
    - Using CSS grid we are going to create rows of 3 mat-card’s
      * grid-container { display: grid; grid-gap: 20px; grid-template-columns: auto auto auto; padding: 25}
      * grid-container\_\_item { padding: 20px; font-size: 30px;}
      * grid-container\_\_ahref:hover { text-decoration: underline; }
      * grid-container\_\_ahref { text-decoration: none; cursor: pointer;}
    - Add a div and give it the class: grid-container
    - Add an ng-container and an \*ngFor loop to loop over the books array
    - In the body of the ng-container element create a mat-card, mat-card-title, and mat-card-content
    - Give the mat-card-title an href with a (click) directive that calls the showBookDetails function passing-in the book.isbn value.  Give the anchor tag the value of book.title
    - In the body of the mat-card-content, create a mat-list with three mat-list-items and map the isbn, pages, and authors to each mat-list-item
      * <mat-list-item>ISBN: {{ book.isbn }}></mat-list-item>
  + Note: we are basically wanting to display a spinner when our books array has a 0 count.  When the books array has at least one object, show the cards.  We are doing this so users are not staring at a blank page while the books are being returned from the Open Library API
* Run and test the application
  + You are verifying the books appear in the book-list.component.html page as a 3 column grid of mat-cards.  When a user selects the card’s title the mat-dialog should open with the selected books details
  + Note: there is one small bug in the application; when you open the last row of records, a small portion of the background color turns from dark grey to white.  **For an extra 20 points**, resolve this minor bug
* Deploy the application to GitHub pages (use the deployment guide I provided under the weekly resources section)
* Update your personal portfolio with a link to the deployed website on GitHub pages
  + Clicking the link in your personal portfolio should take users to the gpa-calculator-app running as a static website on GitHub pages

Note: this week’s exercise is all about updating our existing codebase to use API calls to populate our data.