Author: Roz Pontiff

Version: 1.0.0

Latest version: 12/27/2018 12:04 PM

**Groodle documentation:**

“\*\*\*” indicates that the information needs to be found or checked.

**home.component**

This is the view for our project. It is "where the action occurs." Everything that is visible is there because it is displayed through the html scripting of this component.

**home.component.html**

This is the template for our home component. It contains the HTML code that prints to the screen. It contains three forms. When the user interacts with the forms, it stores their choice in the student model found in student.model.ts.

1) First Name

- adds the student’s first name to the ‘firstName’ field in the student model.

2) Last Name

- adds the student’s last name to the ‘lastName’ field in the student model.

3) Courses Taken

- adds the courses chosen to the ‘coursesTaken’ array in the student model.

4) Submit button (placeholder)

- this lacks functionality right now but will ultimately send the user model to the backend when the user clicks it.

5) Card

- this prints the user’s class choices to the screen with a description of the class when they click on an option in the Courses Taken form. The user can hold down the CTRL button to select more than one class.

[**home.component.ts**](http://home.component.ts)

This is the home.component class file. It contains the fields that store the information contained in the home component and the methods utilized by the home component.

- Imports allow this component to use and interact with other components.

1) **@Component decorator**

- This is what designates the component as a component, and the needed information to display and interact with this component.

a) selector: 'home'

- This provides the “tag” that other components use to interact with it. For example, another component can include “<home></home>” in its HTML file to display this component’s HTML formatting. \*\*\*

b) styleUrls: ['./home.component.css']

- References the style formatting found in './home.component.css', which is currently empty.

c) templateUrl: './home.component.html'

- References the template that displays the view found in './home.component.html'

. This is what displays our home component information to the screen.

2) **export class HomeComponent {**

**-** The “export” keyword is what allows this component to be referenced by other components.

3) **class variables:**

- courses = the array of course objects – initially empty.

- model = the Student model – initially empty.

- hasCoursesTakenError = \*\*\*

4) **constructor**

**-** creates the home component with the two services it uses as the arguments:

- FormPoster – sends the student model to the back end server.

- CourseService – gets the Course List from the back end server.

- calls the getCourses() method, prints success or failure to the console, then prints the courses to the console.

**5) submitForm**

**-** calls ‘validateCoursesTaken()’’ method to determine if the user has chosen any courses.

Return an error if not.

- calls the formPoster component’s ‘postStudentForm’ method to send the student model to the back end server.

6) \*\*\* validateCoursesTaken

**Models Component**

This component contains the class files for both the Course and Student models.

**Course.models.ts**

Contains the class file for the Course model.

Variables: (self explanatory)

public course\_name: string,

public credit\_hours: number,

public course\_description: string,

public speciality: string

**Student.models.ts**

Contains the class file for the Student model.

Variables: (self explanatory)

public firstName: string,

public lastName: string,

public coursesTaken: string[],

**Services Component**

This component contains the class files for both the Course and formPoster services. The extractData() and handleError() methods are standard.

**Course.service.ts**

private extractData()

argument: res: Response

returns: body || { };

private handleError()

argument: error: any

returns:

public getCourses()

argument:

returns: Course list from back end

\*\* not certain what “:Observable<Course[]>” indicates.

**formPoster.service.ts**

private extractData()

argument: res: Response

returns: body || { };

private handleError()

argument: error: any

returns:

postStudentForm(student: Student): Observable<any>

argument: student: Student

returns: the student model

**App Component**

The app component is the Component that “boostraps” the other components. It is the first to run. It chooses the initial view and loads the other components.

**app.component.html**

This is what is initially displayed on the screen.

“<main>

<home></home>

</main>”

Using the home selector means that it will display everything within the HTML template of the files referenced by home selector in the decorator of home.component.ts, which is of course the [home.component.html](http://home.component.html) file. This allows our app to be printed to screen.

**app.component.ts**

Contains the class file for the App component.

“templateUrl: './app.component.html', “- this is what directly displays our initial view by referencing the HTML element that contains our home template.

**app.module.ts**

This contains the “metadata” for our application, ie “how the pieces of your application fit together and what other files and libraries the app requires.”

First, it imports all of the different components, modules, and services that will be used by our app.

Second, it creates the “NgModule” decorator, which declares everything you create and groups them together.

There are four parts to our NgModule decorator:

1) Declarations – this lists everything we will use in our templates. Components are automatically added here when you create a new component using the Angular CLI.

2) Imports – All modules that will be used from outside of your Angular project. This is how we utilize core Angular functionality, much like imports in Java.

3) Providers – this is a list of the services that will be utilized by this module.

4) Bootstrap – this is the root component that Angular creates and inserts into the index.html host web page, aka what is actually displayed at the very beginning. Literally “bootstraps” our program.

**app.routes.ts**

Routing allows us to navigate through our app without switching URLs, which makes things easier and simpler. I don’t think we are using routing right now, but we are considering using it to switch to the results page after the user hits “ok.”