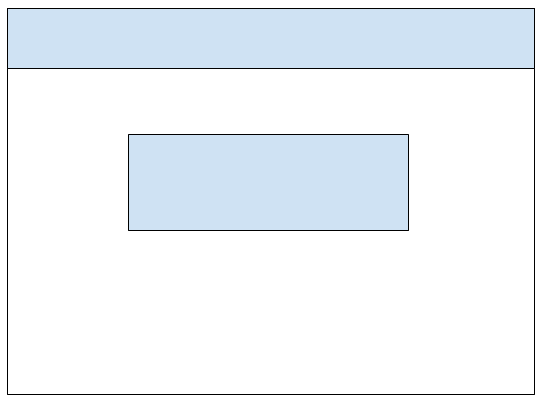
# Instructions - Exercise 3.2 – Passing Data to Routes, Part 1

**Layout**

Composer Details



composer-app, part 2

**Instructions**

* ~~Make a copy of the composer app from Assignment 2.4 and add it to your week-3 directory~~
* ~~Rename the application to enhanced-composer-app~~
* ~~Delete the node\_modules directory~~
* ~~Delete the package-lock.json file~~
* ~~Open the angular.json file and find and replace all “composer-app” entries with “enhanced-composer-app”~~
* ~~Open the package.json file and change the name to “enhanced-composer-app”~~
* ~~Run npm install and ng serve~~
  + ~~You are doing this to test the application and confirm there are no errors~~
* ~~app.component.ts~~
  + ~~Change the assignment’s name to “Exercise 3.2 - Passing Data to Routes, Part 1~~
* ~~Generate a new component and name it composer-details~~
* ~~Create a new TypeScript file under the app directory and name it composer.interface.ts~~
* ~~Create a new TypeScript file under the app directory and name it composer.class.ts~~
* ~~Move the Composer class from the composer-list.component.ts to the composer.class.ts file~~
* ~~composer.interface.ts~~
  + ~~Rename the interface to IComposer~~
    - ~~export interface IComposer~~
  + ~~Add a new field called composerId of type number~~
  + ~~Add a new field called fullName of type string~~
  + ~~Add a new field called genre of type string~~
* composer.class.ts
  + ~~Remove the fullName and genre fields~~
  + ~~Remove the fullName and genre fields from the constructor’s parameters~~
  + ~~Remove the code in the body of the constructor~~
  + ~~Add a new field called composers of type Array<IComposer>~~
  + ~~In the class’s constructor, populate the composer’s array with 5 composer objects~~
    - ~~{composerId: 100, fullName: “Wolfgang Amadeus Mozart”, genre: “Classical”}~~
    - ~~You should be able to reuse most of the code from the body of the constructor in the composer-list.component.ts file.~~
  + ~~Create two new functions: getComposers() and getComposer(composerId: number)~~
  + ~~getComposers()~~
    - ~~Return the composers array~~
  + ~~getComposer(composerId: number)~~
    - ~~Loop over the composer array and return the object that matches the passed-in composerId~~
      * ~~for (let composer of this.composers)~~
      * ~~Note: do not use any built-in filter functions for this exercise.  I want you to get comfortable with looping over arrays and extracting objects~~
* composer-list.component.ts
  + ~~Add an import for the IComposer interface~~
    - ~~import { IComposer } from ‘../composer.interface’~~
  + ~~Add an import for the Composer class~~
    - ~~import { Composer } from ‘../composer.class’;~~
  + ~~Update the composer’s variable to be of type Array<IComposer>~~
  + ~~In the components constructor create a new Composer object and assign the getComposers() function to the composers variable~~
    - ~~this.composers = new Composer().getComposers();~~
    - ~~Do not forget to the remove the old code from Assignment 2.4.~~
* ~~composer-list.component.html~~
  + ~~Add a new column to the table for Composer ID~~
  + ~~Update the table definition to include the composerId~~
    - ~~{{ composer.composerId  }}~~
* ~~app-routing.module.ts~~
  + ~~Add a new entry for the composer details page with an id parameter called composerId~~
    - ~~{ path: ‘composer-details/:composerId’ }~~
* ~~composer-list.component.html~~
  + ~~Wrap the composerId table definition with an HTML anchor tag and pass-in the composerId value~~
    - ~~<a routerLink=”/composer-details/{{composer.composerId}}”>{{composer.composerId}}</a>~~
* composer-details.component.ts
  + ~~Add an import statement for the ActivatedRoute from Angular’s built-in RouterModule~~
    - ~~import { ActivatedRoute } from ‘@angular/router’;~~
  + ~~Add the ActivatedRoute object to the component’s constructor~~
    - ~~constructor(route: ActivatedRoute)~~
  + ~~Add a variable called composerId of type number~~
  + ~~In the components constructor and using the ActivatedRoute object, call the snapshot function to retrieve the composerId parameter value and assign it to the composerId variable.  Parameter values will always return a string value, so you will need to use parseInt to parse the string into a numerical value~~
    - ~~this.composerId = parseInt(this.route.snapshot.paramMap.get(‘composerId’), 10);~~
  + ~~Add an import statement for the IComposer interface~~
    - ~~import { IComposer } from ‘../composer.interface’;~~
  + ~~Add a variable called composer of type IComposer~~
  + ~~Add an import statement for the Composer class~~
  + ~~In the body of the component’s constructor (under the code for the composerId)~~
    - ~~Add an if statement to check the composerId value~~
      * ~~If (this.composerId) {}~~
    - ~~In the body of the if statement, create a new Composer object and assign the getComposer(composerId: number) function to the composer variable~~
      * ~~this.composer = new Composer().getComposer(this.composerId);~~
* ~~composer-details.component.html~~
  + ~~Create a Bootstrap list group and output each value of the composer object~~
  + ~~Under the Bootstrap list group add an anchor tag with a routerLink that returns users to the composer-list route.  Float the anchor tag to the right~~
    - ~~<a routerLink=”/composer-list” class=”float-right”>Return to Composers</a>~~
  + ~~Selecting “Return to Composers” should return users to the Composer list page~~
* ~~Run the application and see the results~~
  + ~~You should be able to select the id of a Composer record and view its details~~