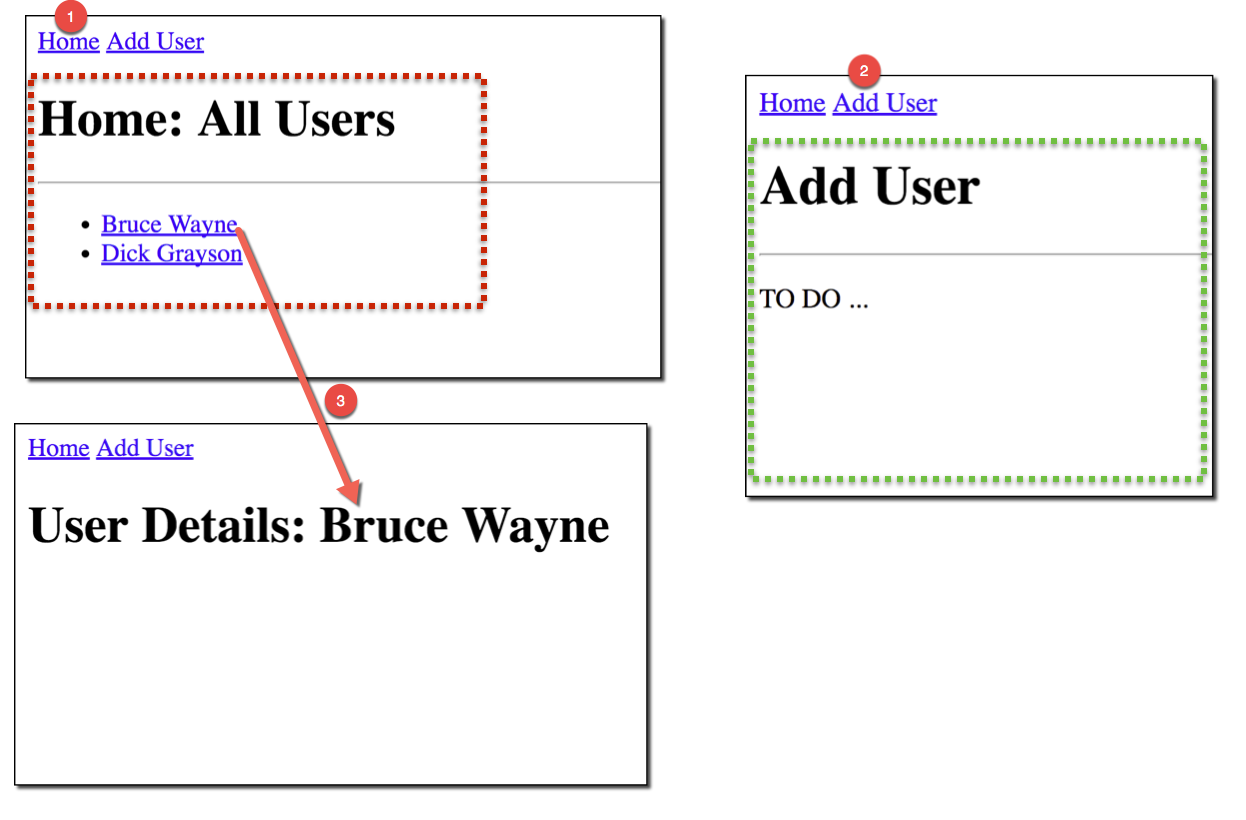
Add Navigation Routes Lab

Angular supports navigation routes. You can create links in your app to route to different views.

In this lab you create an application with the following flow:



|  |  |
| --- | --- |
|  | Any time you create or modify a file, save it. |

**Goals**

* Create supporting view components
* Configure application routes
* Pass application routes to router module
* Specify the location of rendered content
* Create HTML links on a page
* Set up a routing strategy
* Pass data to routes

1. Set Up the Project

1.1. Create a New Angular Project

In this exercise you use the Angular CLI tool to create a new Angular project.

1. Move into the training directory:

cd angular-training

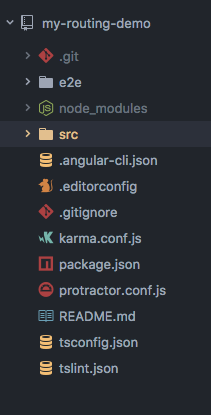
1. Create a new project:

ng new my-routing-demo

|  |  |
| --- | --- |
|  | This command may take several minutes to complete. |

1.2. Add the Angular Project

1. Start the **Atom** text editor.
2. Select **File** → **Add Project Folder**.
3. Navigate to **angular-training/my-routing-demo** and select **Open**.
   * Your screen should look like this:



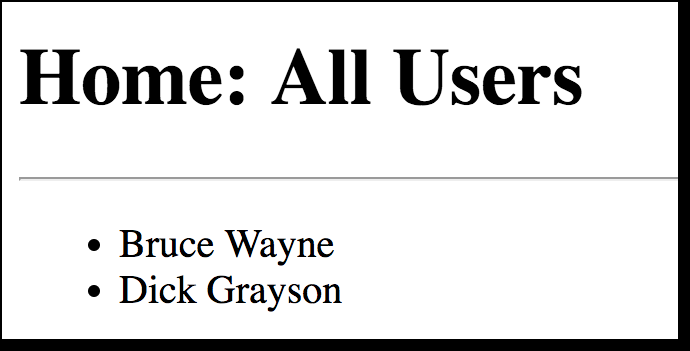
2. Create Supporting View Components

These exercises create view components for routing.

2.1. Create the Home Component

In this exercise you create the Home component view. This view displays a list of users. The list of users is hard-coded. Later in the course you retrieve the users from a service.

The final view component looks like this:



1. In the **src/app** directory, create a new file **home.component.ts**.
2. Add the following code to the file:
3. import { Component } from '@angular/core';
4. @Component({
5. selector: 'home',
6. template: `
7. <h1>Home: All Users</h1>
8. <hr>
9. <ul>
10. <li>Bruce Wayne</li>
11. <li>Dick Grayson</li>
12. </ul>
13. `
14. })

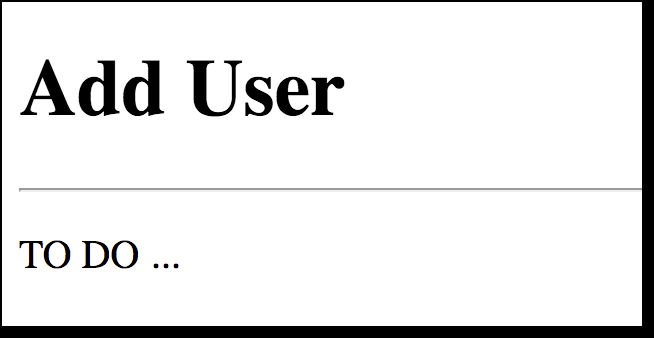
export class HomeComponent { }

* + Instead of being a separate file, the HTML template is defined with inline content that spans multiple lines. TypeScript supports multiline strings delimited with back tick characters.

2.2. Create the Add User Component

In this exercise you create the Add User component view. This view simply displays a header with the text **Add User**. There isn’t a form to use. Forms are covered later in the course. For now, you simply create an additional view component for routing.

Your final view component looks like this:



1. In the **src/app** directory, create a new file **add-user.component.ts**.
2. Add the following code to the file:
3. import { Component } from '@angular/core';
4. @Component({
5. selector: 'add-user',
6. template: `
7. <h1>Add User</h1>
8. <hr>
9. <p>TO DO ...</p>
10. `
11. })

export class AddUserComponent { }

3. Configure the Application Routes

1. In the **src/app** directory, create a new file **app.routing.ts** to define application routes and also configure routes for the root module.
2. Add the following code to the file:
3. import { Routes, RouterModule } from '@angular/router';
4. import { HomeComponent } from './home.component';
5. import { AddUserComponent } from './add-user.component';
6. // define my routes
7. const routes : Routes = [
8. { path : '', component : HomeComponent}, // default route
9. { path : 'add-user', component : AddUserComponent}
10. ];
11. // configure routes for root module

export const routing = RouterModule.forRoot(routes);

* + At the beginning of the file, components are imported.
  + Then routes are defined and associated with the components:
    - The default route renders **HomeComponent**.
    - The **add-user** path renders **AddUserComponent**.

4. Specify Location of Rendered Content

**<router-outlet>** is used to show the rendered content. This area of the page is updated based on route navigation.This area is also where HTML links are created.

1. Open the file **src/app/app.component.ts**:
2. import { Component } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. template: `
6. <a [routerLink]="['/']">Home</a>
7. <a [routerLink]="['/add-user']">Add User</a>
8. <router-outlet></router-outlet>
9. `
10. })

export class AppComponent { }

1. Note the line **<a [routerLink]="['/']">Home</a>**. This is the default link.
2. Note the next line **<a [routerLink]="['/add-user']">Add User</a>**. This link sends the user to the **add-user** route.

5. Set Up the Routing Strategy

1. Open the file **src/app/app.module.ts**.
2. Move to the **imports** section and add the following:
3. import { routing } from './app.routing';
4. import { LocationStrategy, HashLocationStrategy } from '@angular/common';
5. import { AppComponent } from './app.component';
6. import { HomeComponent } from './home.component';

import { AddUserComponent } from './add-user.component';

* + The first line imports the array of routes defined in **app.routing.ts**.
  + The second line defines the classes for the hash-based routing strategy.
  + In the third line, **AppComponent** is the main component for the application.
  + The fourth and fifth lines are the custom component views created in this lab.

1. Move down to the **declarations** section and update it to look like this:
2. declarations: [
3. AppComponent,
4. HomeComponent, <a>
5. AddUserComponent <b>

],

* + **HomeComponent** and **AddUserComponent** are the custom component view created in this lab. This section tells Angular that your app uses these components.

1. Move to the **imports** section and update it to look like this:
2. imports: [
3. BrowserModule,
4. FormsModule,
5. HttpModule,
6. routing <a>

],

* + This section configures the module to use the **routing** variable exported from **app.routing.ts**.

1. Move to the **providers** section and update it as follows:

providers: [ { provide: LocationStrategy, useClass: HashLocationStrategy} ],

* + This configures Angular to use the hash-based location strategy.
  + Your final **app.module.ts** file should should look like this:
  + import { BrowserModule } from '@angular/platform-browser';
  + import { NgModule } from '@angular/core';
  + import { FormsModule } from '@angular/forms';
  + import { HttpModule } from '@angular/http';
  + import { routing } from './app.routing';
  + import { LocationStrategy, HashLocationStrategy } from '@angular/common';
  + import { AppComponent } from './app.component';
  + import { HomeComponent } from './home.component';
  + import { AddUserComponent } from './add-user.component';
  + @NgModule({
  + declarations: [
  + AppComponent,
  + HomeComponent,
  + AddUserComponent
  + ],
  + imports: [
  + BrowserModule,
  + FormsModule,
  + HttpModule,
  + routing
  + ],
  + providers: [ { provide: LocationStrategy, useClass: HashLocationStrategy} ],
  + bootstrap: [AppComponent]
  + })

export class AppModule { }

6. Test Your Application

1. Move back to the **angular-training** terminal window.
2. Move into the project directory:

cd my-routing-demo

1. Run the app:

ng serve

1. Wait until you see the following text:
2. ng serve
3. \*\* NG Live Development Server is running on http:*//localhost:4200 \*\**
4. Hash: 719df4eb209182a9a4fc
5. Time: 10090ms
6. chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 157 kB {4} [initial] [rendered]
7. chunk {1} main.bundle.js, main.bundle.js.map (main) 4.05 kB {3} [initial] [rendered]
8. chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 9.77 kB {4} [initial] [rendered]
9. chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.69 MB [initial] [rendered]
10. chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]

webpack: Compiled successfully.

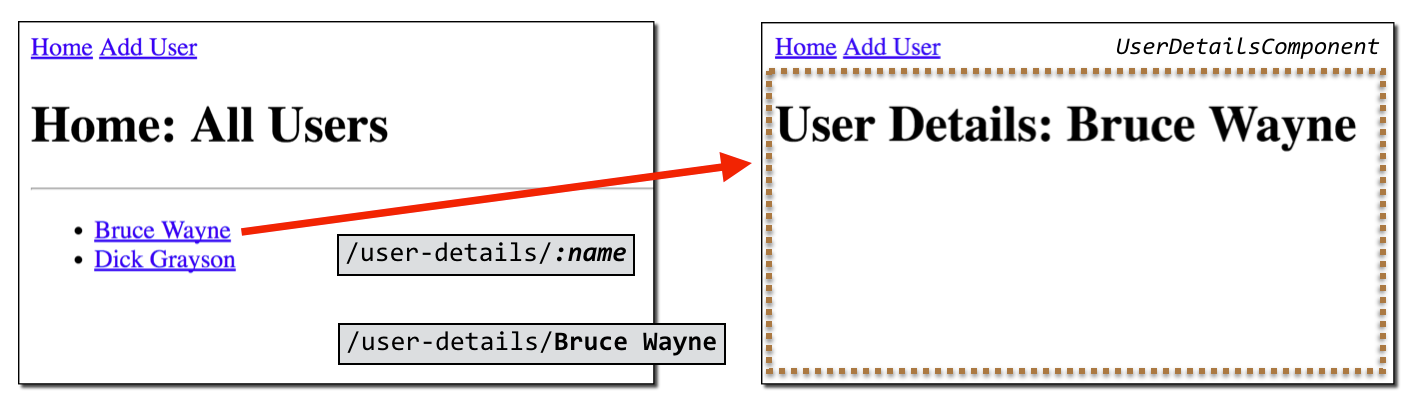
1. Keep this terminal window running. It contains the server that is listening on port 4200.
2. Click this link to go to [http://localhost:4200](http://localhost:4200/).
   * You should see a screen similar to this:



1. Test the links.

7. Pass Data to the Routes

In this exercise you update your application to pass data to routes. The list of users contains links. The links send the user’s name to the user details view component. The application interaction is shown below:



7.1. Define the App Routing

Add a new route for **users-details** that has a parameter for **name**.

1. Open **src/app/app-routing.ts**.
2. Add the imports for **UserDetailsComponent**. (You create this component in the next exercise.)

import { UserDetailsComponent } from './user-details.component';

1. In the declaration for the **routes** constant, add a comma to the end of this line of code:

{ path : 'add-user', component : AddUserComponent},

1. On the next line, add the following code:

{ path : 'user-details/:name', component : UserDetailsComponent}

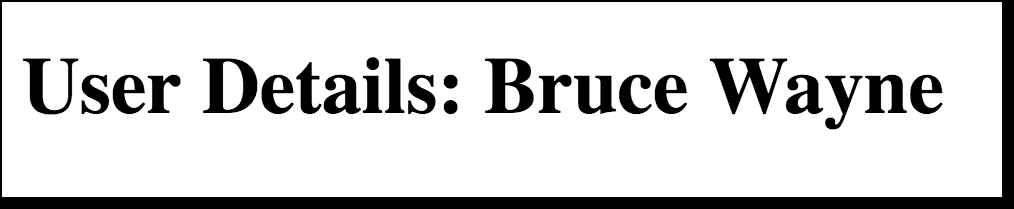
* + The file should look like this:
  + import { Routes, RouterModule } from '@angular/router';
  + import { HomeComponent } from './home.component';
  + import { AddUserComponent } from './add-user.component';
  + import { UserDetailsComponent } from './user-details.component';
  + // define my routes
  + const routes : Routes = [
  + { path : '', component : HomeComponent},
  + { path : 'add-user', component : AddUserComponent},
  + { path : 'user-details/:name', component : UserDetailsComponent}
  + ];
  + // configure routes for root module

export const routing = RouterModule.forRoot(routes);

7.2. Create the **UserDetails** Component

In this exercise you create the **UserDetails** component view. This view displays the user’s name. The user’s name is sent as a parameter to the view component.

Your final view component looks like this:



1. In the **src/app** directory, create a new file **user-details.component.ts**.
2. Add the following code to the file:
3. import { Component } from '@angular/core';
4. import { ActivatedRoute } from '@angular/router';
5. @Component({
6. selector: 'user-details',
7. template: `
8. <h1>User Details: {{userInfo}}</h1> <a>
9. `
10. })
11. export class UserDetailsComponent {
12. userInfo : string; <a>
13. constructor(route : ActivatedRoute) {
14. this.userInfo = route.snapshot.params["name"]; <b>
15. }

}

* + **{{userInfo}}** binds **userInfo** into the HTML template.
  + In the constructor (bottom of the file), the component reads the **name** parameter and assigns it to the **userInfo** field.

7.3. Add Links to Home Component

1. Open **src/app/home.component.ts**.
2. Add an import statement for the **Router** class:

import { Router } from '@angular/router';

1. Add links for each user:
2. <li><a [routerLink]="['/user-details', 'Bruce Wayne']">Bruce Wayne</a></li>

<li><a [routerLink]="['/user-details', 'Dick Grayson']">Dick Grayson</a></li>

* + The parameter values are hard coded. Later in the course you make them dynamic.
  + Your final code should look like this:
  + import { Component } from '@angular/core';
  + import { Router } from '@angular/router';
  + @Component({
  + selector: 'home',
  + template: `
  + <h1>Home: All Users</h1>
  + <hr>
  + <ul>
  + <li><a [routerLink]="['/user-details', 'Bruce Wayne']">Bruce Wayne</a></li>
  + <li><a [routerLink]="['/user-details', 'Dick Grayson']">Dick Grayson</a></li>
  + </ul>
  + `
  + })

export class HomeComponent { }

* + This code passes the user’s name to the route **user-details**.

7.4. Update the Module

1. Open **src/app/app.module.ts**.
2. Move to the **imports** section and add the following:

import { UserDetailsComponent } from './user-details.component';

* + This imports the component just created for the User Details view.

1. Move down to the **declarations** section and add **UserDetailsComponent** to it:
2. declarations: [
3. AppComponent,
4. HomeComponent,
5. AddUserComponent,
6. UserDetailsComponent

],

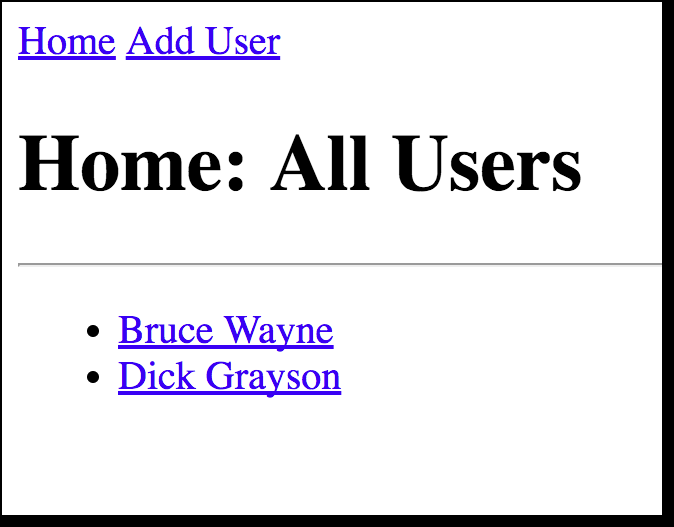
* + This is the new user details component view.

8. Test Your Application

1. Make sure all of your files are saved.

|  |  |
| --- | --- |
|  | The **ng serve** command should still be running. It automatically recompiles your code for you and refreshes the browser. |

1. Move to your web browser. You should see a screen similar to this:



1. Click the links. You should see the following application flow:

|  |
| --- |
|  |

