

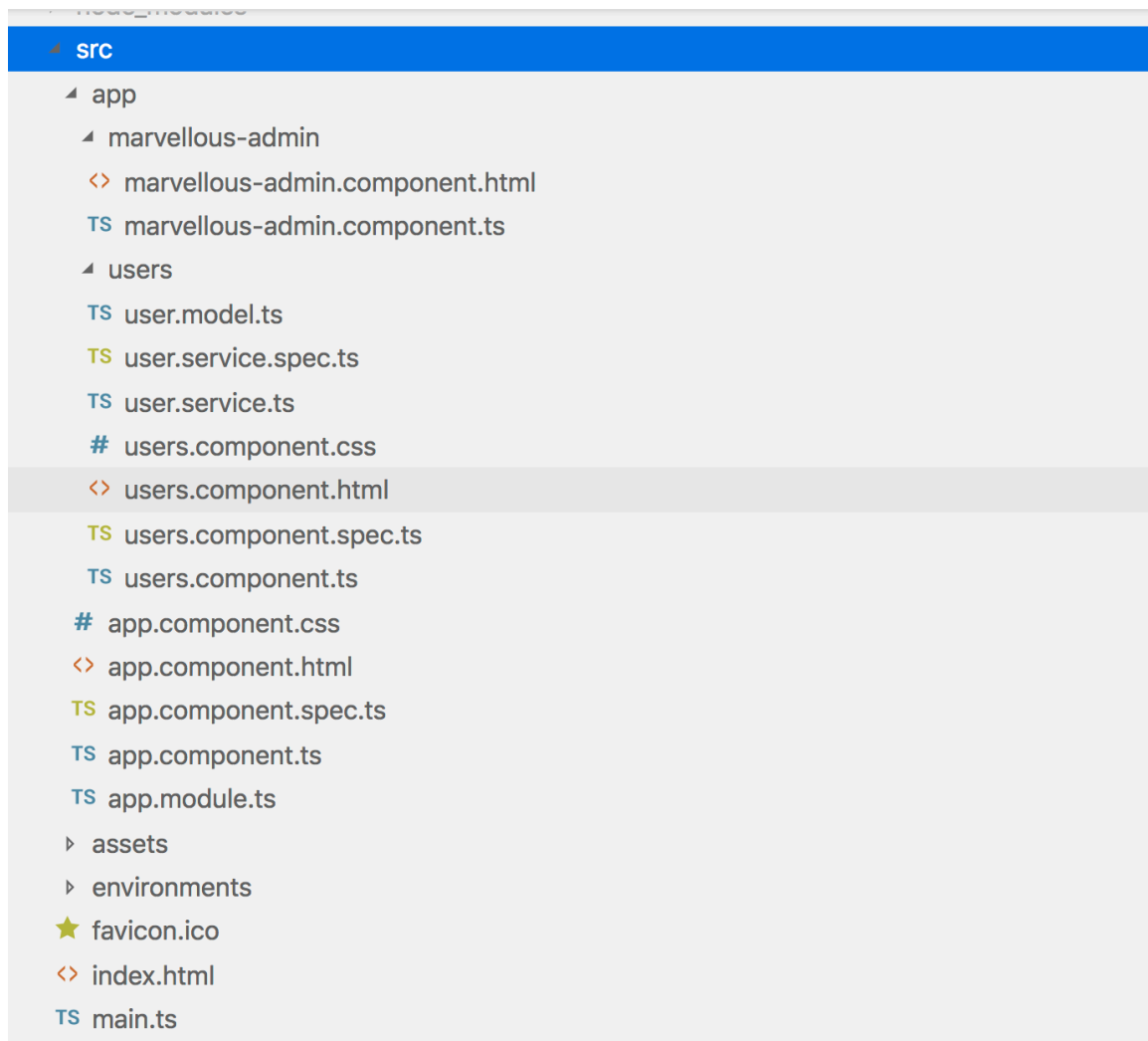
Augury

- Augury is a tool designed specifically for Angular apps.
- It's an open-source debugging and profiling tool for Angular 7 applications.
- Augury is just a Chrome extension that's quite simple to use.
- Augury visualizes our app's structure in a form of a tree, showing how components and their dependencies relate to each other.
- It also allows us to inspect properties of our objects and change them on the fly.
- On top of that, we can easily view the source code of a specific component, insert breakpoints as needed, work with events, and more.
- Lastly, We can browse the application's routing system, as well as view the full list of all utilized modules.
- We can install Augury from <https://augury.rangle.io/>
- After that, we may open the developer tools by pressing Ctrl + Shift + I (Windows/Linux) or Cmd + Opt + I (macOS).
- On browser new tab called Augury has appeared.
- After switching to this tab, we will either see the application's structure or the phrase "This application is not an Angular application".

Consider our Angular application MarvellousAugury to understand working of Augury.

About **MarvellousAugury** Application :

Consider below project layout of our application.



- This application contains two custom components as MarvellousAdmin & User.
- There is one model named as User model. Which contains one class which maintains information about user.

export class User

```
{  
  id: number = 0;  
  first: string = '';  
  last: string = '';  
}
```

- There is one user defined service named as UserService. This service return a list of hardcoded users.

import { Injectable } from '@angular/core';

import { User } from './user.model';

```
@Injectable()  
export class UserService  
{
```

```
  constructor() { }
```

```
  getUsers(): User[] {  
    return [
```

```
      {  
        id: 1,  
        first: 'Piyush',  
        last: 'Khairnar'  
      },
```

```
      {  
        id: 2,  
        first: 'Pooja',  
        last: 'Khairnar'  
      },
```

```
      {  
        id: 3,  
        first: 'Madhavi',  
        last: 'Khairnar'  
      }  
    ]  
  }  
}
```

- we allowing the users to be selected by clicking on them. The currently selected user is stored in a separate selectedUser attribute.

```
<div *ngFor="let user of users" (click)="onSelect(user)"  
[class.selected]="user === selectedUser">  
  <p>  
    {{user.last}}, {{user.first}} (ID: {{user.id}})  
  </p>  
</div>
```

```
<div *ngIf="selectedUser">
```

```
  <h3>Edit Information of Marvellous Users</h3>
```

```
  <label for="first"> First Name </label>
```

```
  <input [(ngModel)]="selectedUser.first" placeholder="First name" id="first">
```

```
  <label for="last"> Last Name </label>
```

```
  <input [(ngModel)]="selectedUser.last" placeholder="Last name" id="last">
```

```
</div>
```

- We access userService from User component using the dependency injection mechanism.

```
import { Component, OnInit } from '@angular/core';
```

```
import { User } from './user.model';
```

```
import { UserService } from './user.service';
```

```
@Component({  

  selector: 'app-users',  

  templateUrl: './users.component.html',  

  styleUrls: ['./users.component.css']  

})  

export class UsersComponent implements OnInit  

{  

  users: User[];  

  selectedUser: User;  

  onSelect(user: User): void  

  {  

    this.selectedUser = user;  

  }  

  constructor(private userService: UserService) { }  

  ngOnInit()  

  {  

    this.getUsers();  

  }  

  getUsers(): void  

  {  

    this.users = this.userService.getUsers();  

  }  

}
```

- Both the components User and MarvellousAdmin are rendered from app component.

```
<div style="text-align:center">
```

```
  <h1>  

    Marvellous Infosystems  

  </h1>
```

```
  <h2>  

    Augury Demonstration  

  </h2>
```

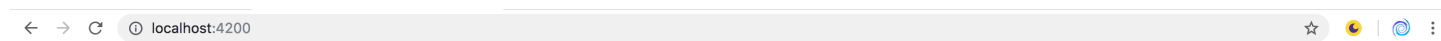
```
</div>
```

```
<app-marvellous-admin></app-marvellous-admin>
```

```
<h3>Users of Marvellous Infosystems Admin panel</h3>
```

```
<app-users></app-users>
```

Our application looks like



Marvellous Infosystems

Augury Demonstration

Inside Marvellous Admin Panel

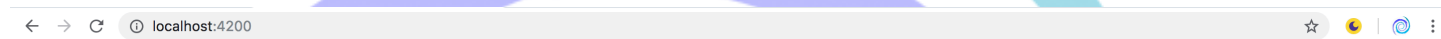
Users of Marvellous Infosystems Admin panel

Khairnar, Piyush (ID: 1)

Khairnar, Pooja (ID: 2)

Khairnar, Madhavi (ID: 3)

When we select specific user then it displays its edit option as



Marvellous Infosystems

Augury Demonstration

Inside Marvellous Admin Panel

Users of Marvellous Infosystems Admin panel

Khairnar, Piyush (ID: 1)

Khairnar, Pooja (ID: 2)

Khairnar, Madhavi (ID: 3)

Edit Information of Marvellous Users

First Name Last Name

Now we use above MarvellousAugury Application to demonstrate features of Augury

For component view Press **Ctrl + Shift + I** or **Cmd + Opt + I**, switch to the Augury tab, and press AppComponent in the left pane, under the Component Tree

We can also change the contents from Augury panel so that changes gets reflected in the targeted output.

When we click one view source link it displays source code of our component.

When we select User component then we get all dependencies and values from the array which is returned by the service.

Marvellous Infosystems Pune

Augury Demonstration

Inside Marvellous Admin Panel

Users of Marvellous Infosystems Admin panel

Khairnar, Piyush (ID: 1)	Us
Khairnar, Pooja (ID: 2)	
Khairnar, Madhavi (ID: 3)	

Component Inspector (UsersComponent):

- State:**

```

userService: {}
users: Array(3)
  0: Object
    id: 1
    first: Piyush
    last: Khairnar
  1: Object
    id: 2
    first: Pooja
    last: Khairnar
  2: Object
    id: 3
    first: Madhavi
    last: Khairnar

```
- Dependencies:**
 - UserService

From the source code of user component we can apply breakpoint by clicking on left side of the line.

Marvellous Infosystems Pune

Augury Demonstration

Inside Marvellous Admin Panel

Users of Marvellous Infosystems Admin panel

Khairnar, Piyush (ID: 1)	Us
Khairnar, Pooja (ID: 2)	
Khairnar, Madhavi (ID: 3)	

Source Code (users.component.ts):

```

1 import { Component, OnInit } from '@angular/core';
2 import { User } from './user.model';
3 import { UserService } from './user.service';
4
5 @Component({
6   selector: 'app-users',
7   templateUrl: './users.component.html',
8   styleUrls: ['./users.component.css']
9 })
10 export class UsersComponent implements OnInit {
11   users: User[];
12   selectedUser: User;
13
14   onSelect(user: User): void {
15     {
16       this.selectedUser = user;
17     }
18   }
19 }
20
21 constructor(private userService: UserService) { }
22
23 ngOnInit() {
24   this.getUsers();
25 }
26
27 getUsers(): void {
28   {
29     this.users = this.userService.getUsers();
30   }
31 }
32 }
33

```

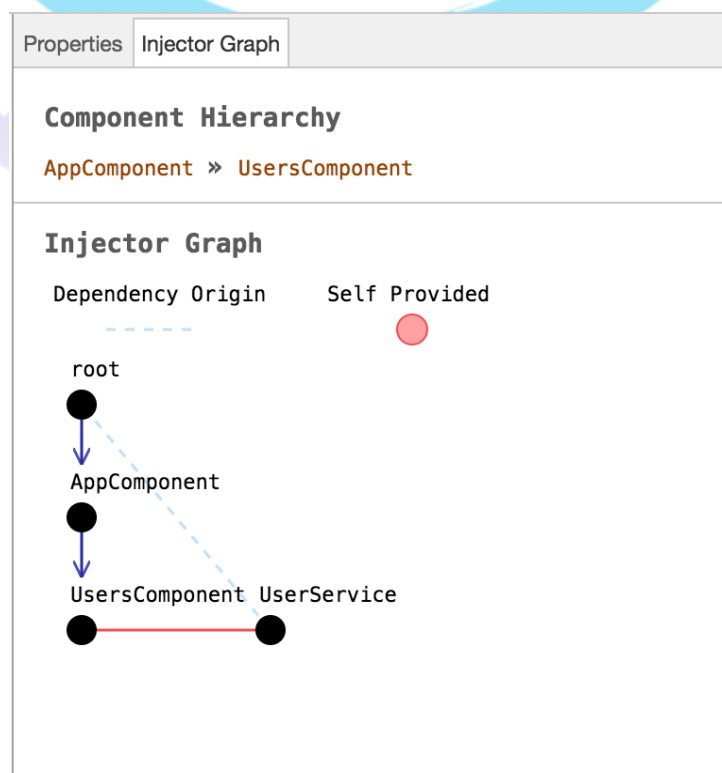
Breakpoints:

- users.component.ts:18: this.selectedUser = user;

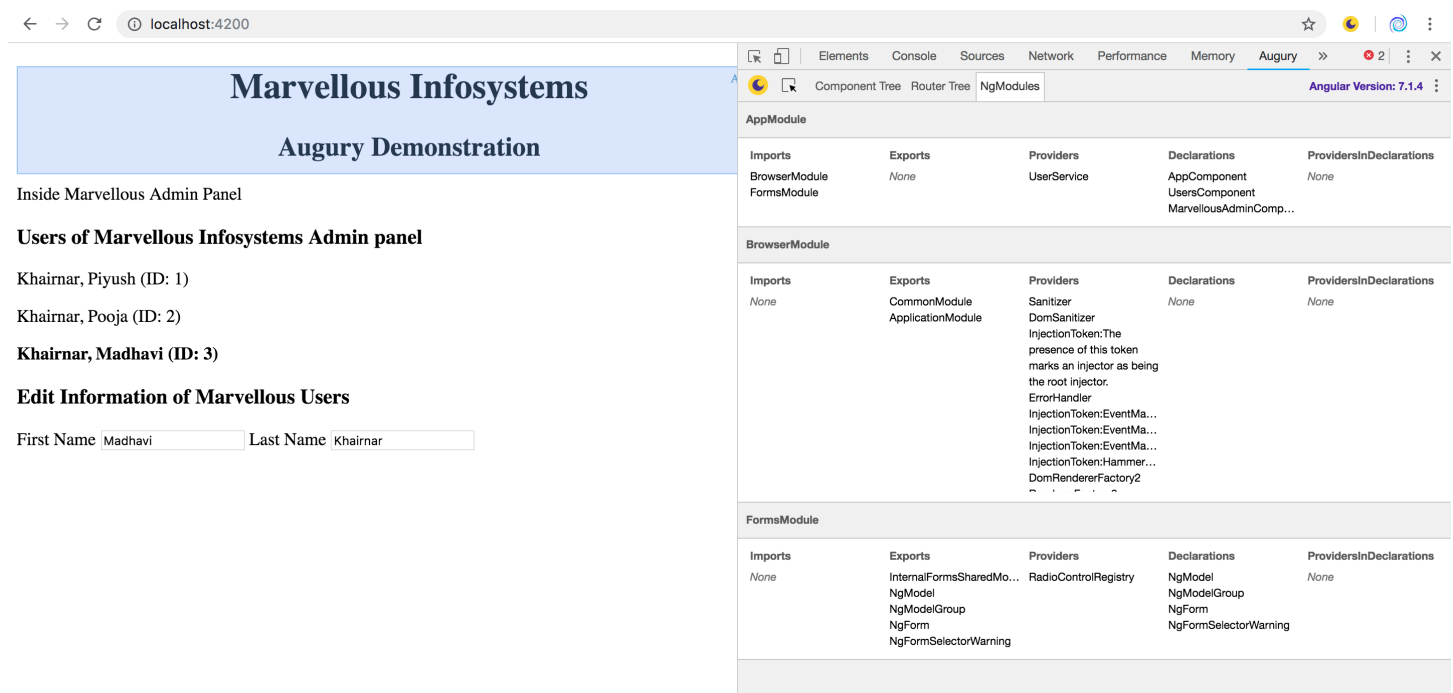
After applying the breakpoint when we click specific user then our code execution automatically stops at the breakpoint.

The screenshot shows a web application running in a browser at localhost:4200. The application displays the 'Marvellous Infosystems Augury Demonstration' and a list of users: 'Khairnar, Piyush (ID: 1)', 'Khairnar, Pooja (ID: 2)', and 'Khairnar, Madhavi (ID: 3)'. The browser's developer tools are open, showing the 'Sources' tab. The code is paused at line 18 of 'users.component.ts', which is 'this.selectedUser = user;'. The console shows the call stack, and the scope shows the current state of the component and the selected user.

As we inject User service into user component we can get that information by selecting injector graph of Augury.



We can also check list of all used modules of our application by selecting ngModule tab.



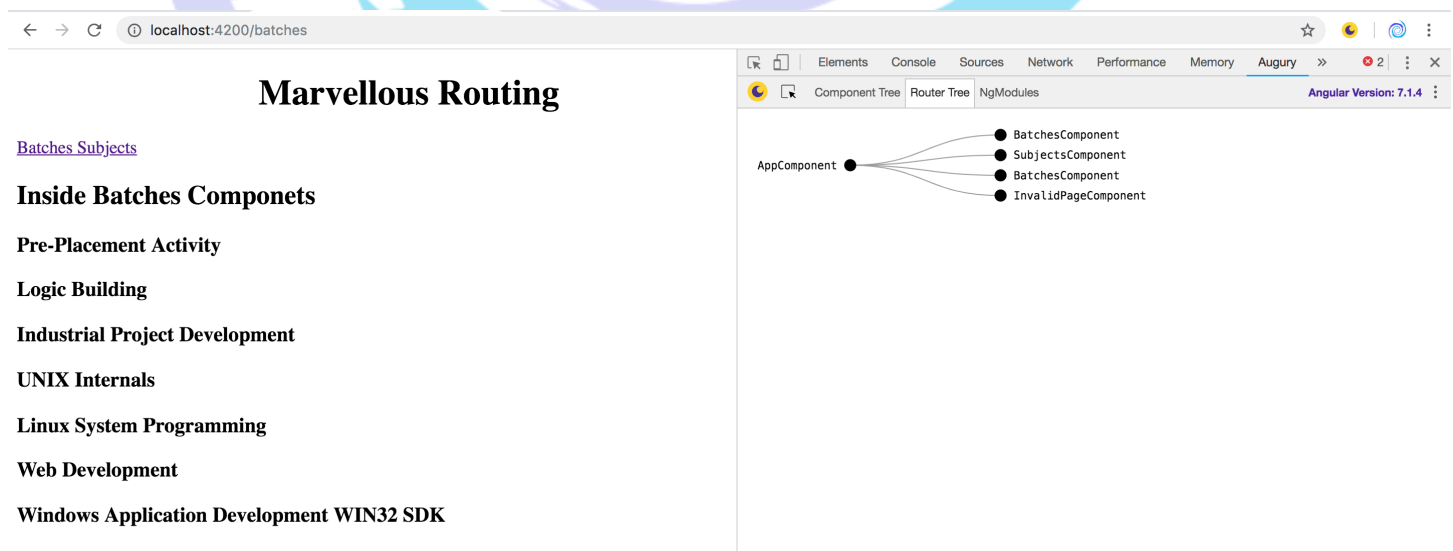
The screenshot shows a web browser at localhost:4200 displaying the Marvellous Admin Panel. The panel has a header with the Marvellous Infosystems logo and the title 'Augury Demonstration'. Below the header, there is a section 'Inside Marvellous Admin Panel' and a section 'Users of Marvellous Infosystems Admin panel' listing three users: Khairnar, Piyush (ID: 1), Khairnar, Pooja (ID: 2), and Khairnar, Madhavi (ID: 3). There is also a section 'Edit Information of Marvellous Users' with input fields for First Name and Last Name.

On the right side of the browser, the Angular Augury NgModules tab is open, showing a tree view of the application's modules. The tree is organized into three main sections: AppModule, BrowserModule, and FormsModule. Each section has a table with columns: Imports, Exports, Providers, Declarations, and ProvidersInDeclarations.

Imports	Exports	Providers	Declarations	ProvidersInDeclarations
BrowserModule FormsModule	None	UserService	AppComponent UsersComponent MarvellousAdminComp...	None
None	CommonModule ApplicationModule	Sanitizer DomSanitizer InjectionToken:The presence of this token marks an injector as being the root injector. ErrorHandler InjectionToken:EventMa... InjectionToken:EventMa... InjectionToken:EventMa... InjectionToken:Hammer... DomRendererFactory2	None	None
None	InternalFormsSharedMo... NgModel NgModelGroup NgForm NgFormSelectorWarning	RadioControlRegistry	NgModel NgModelGroup NgForm NgFormSelectorWarning	None

If our application contains routes then using augury we can get the information about all the routes.

As there is no route in our application we use another Angular application as MarvellousWildcards which contains 4 routes.



The screenshot shows a web browser at localhost:4200/batches displaying the Marvellous Routing application. The application has a header with the title 'Marvellous Routing' and a list of links: Batches Subjects, Inside Batches Componets, Pre-Placement Activity, Logic Building, Industrial Project Development, UNIX Internals, Linux System Programming, Web Development, and Windows Application Development WIN32 SDK.

On the right side of the browser, the Angular Augury Router Tree tab is open, showing a tree view of the application's routes. The tree is organized into a single section: AppComponent. The AppComponent has four children: BatchesComponent, SubjectsComponent, BatchesComponent, and InvalidPageComponent.

```

graph LR
    AppComponent --> BatchesComponent1[BatchesComponent]
    AppComponent --> SubjectsComponent
    AppComponent --> BatchesComponent2[BatchesComponent]
    AppComponent --> InvalidPageComponent
  
```