



Codergirl – Frontend

Unit 2 - Class 7

December 19, 2020

Agenda

- Section 11 - Routing
- *Holiday break is 12/28 to 12/30, with the last class being 12/23 and the first of new year being on 1/4/21*

Routing

- Separate pages for servers, users, home
- Having separate pages and routing to separate pages.
- /users goes to users component
- /servers goes to servers component
- / goes to home component
- App.modules.ts about routes (appRoutes)

Routing – app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { UsersComponent } from './users/users.component';
import { ServersComponent } from './servers/servers.component';
import { UserComponent } from './users/user/user.component';
import { EditServerComponent } from './servers/edit-server/edit-server.component';
import { ServerComponent } from './servers/server/server.component';
import { ServersService } from './servers/servers.service';
import { RouterModule, Routes } from '@angular/router';

const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'users', component: UsersComponent},
  { path: 'servers', component: ServersComponent}
];

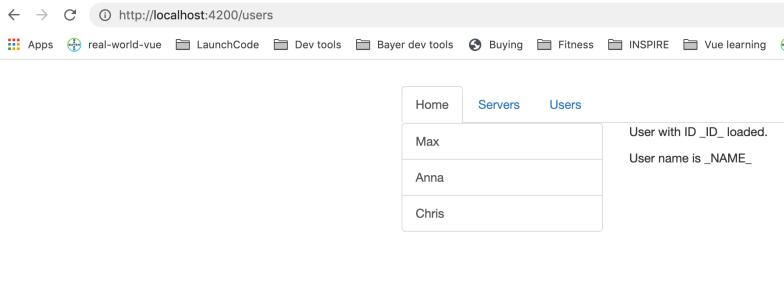
@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    UsersComponent,
    ServersComponent,
    UserComponent,
    EditServerComponent,
    ServerComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    RouterModule.forRoot(appRoutes)
  ],
  providers: [ServersService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Routing – app.component.html

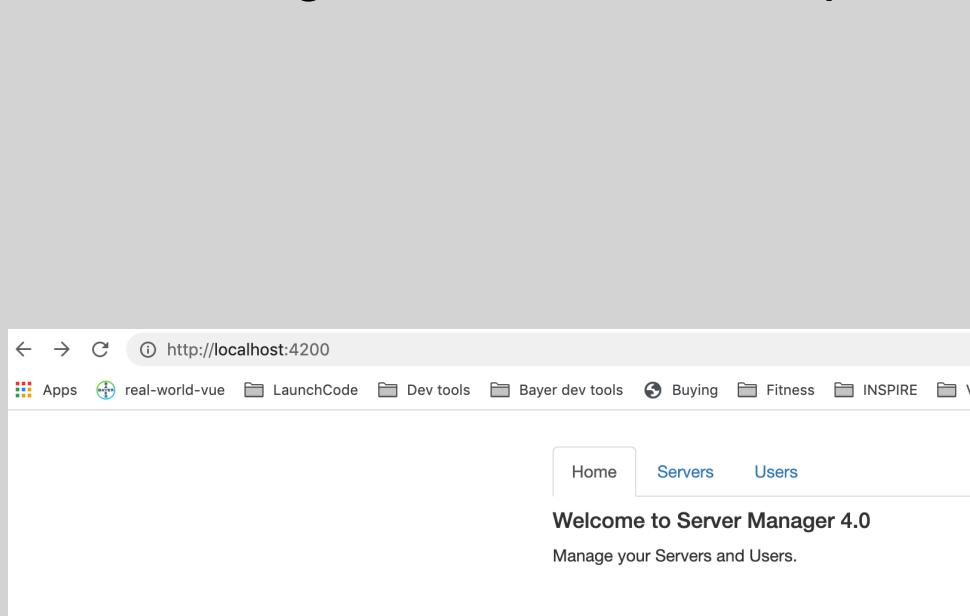
```
<div class="container">
<div class="row">
  <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
    <ul class="nav nav-tabs">
      <li role="presentation" class="active"><a href="#">Home</a></li>
      <li role="presentation"><a href="#">Servers</a></li>
      <li role="presentation"><a href="#">Users</a></li>
    </ul>
  </div>
</div>
<div class="row">
  <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
    <router-outlet></router-outlet>
  </div>
</div>
</div>
```

Test site

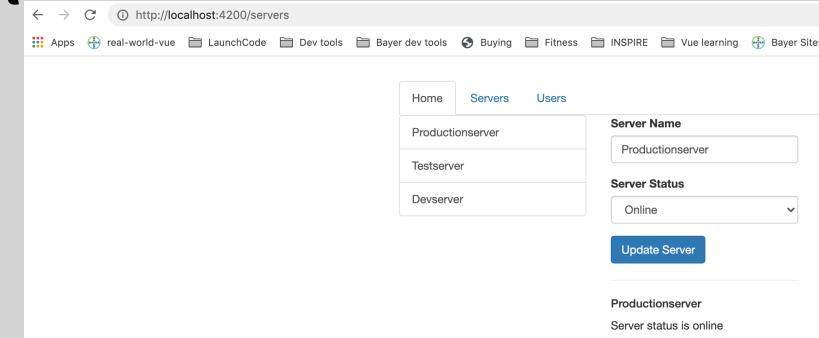
- /users goes to users component



- /servers goes to servers component



Works but app reloads



Fix links

```
<div class="container">
  <div class="row">
    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
      <ul class="nav nav-tabs">
        <li role="presentation" class="active"><a href="/">Home</a></li>
        <li role="presentation"><a href="/servers">Servers</a></li>
        <li role="presentation"><a href="/users">Users</a></li>
      </ul>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
      <router-outlet></router-outlet>
    </div>
  </div>
</div>
```

Works but app reloads

Navigation – user directive instead

- app-component.html. Works without reload.

```
<div class="container">
  <div class="row">
    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
      <ul class="nav nav-tabs">
        <li role="presentation" class="active"><a routerLink="/">Home</a></li>
        <li role="presentation"><a routerLink="/servers">Servers</a></li>
        <li role="presentation"><a [routerLink]=["/users"]>Users</a></li>
      </ul>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
      <router-outlet></router-outlet>
    </div>
  </div>
</div>
```

Works without the /servers with servers as well.

Navigation paths

- /servers Absolute path
- Relative path servers or ./servers or ../servers

Active Router Links

- /servers Absolute path
- Relative path servers or ./servers or ../servers

```
<div class="container">
  <div class="row">
    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
      <ul class="nav nav-tabs">
        <li role="presentation" routerLinkActive="active"
            [routerLinkActiveOptions]="{exact: true}">
          <a routerLink="/">Home</a></li>
        <li role="presentation" routerLinkActive="active"><a routerLink="servers">Servers</a></li>
        <li role="presentation" routerLinkActive="active"><a [routerLink]=["'users'"]>Users</a></li>
      </ul>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
      <router-outlet></router-outlet>
    </div>
  </div>
</div>
```

Navigating Programmatically

```
<h4>Welcome to Server Manager 4.0</h4>
<p>Manage your Servers and Users.</p>
<button class="btn btn-primary" (click)="onLoadServers()">Load Servers</button>
```

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit() {
  }

  onLoadServers() {
    this.router.navigate(['/servers']);
  }
}
```

Relative paths for Navigating Programmatically

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        href="#"
        class="list-group-item"
        *ngFor="let server of servers"
        {{ server.name }}>
        </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr>
    <app-server></app-server>
  </div>
</div>
```

Relative paths for Navigating Programmatically

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { ServersService } from './servers.service';

@Component({
  selector: 'app-servers',
  templateUrl: './servers.component.html',
  styleUrls: ['./servers.component.css']
})
export class ServersComponent implements OnInit {
  public servers: {id: number, name: string, status: string}[] = [];

  constructor(private serversService: ServersService,
              private router: Router) { }

  ngOnInit() {
    this.servers = this.serversService.getServers();
  }
  onReload() {
    this.router.navigate(['servers'])
  }
}
```

Relative paths for Navigating Programmatically

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { ServersService } from './servers.service';

@Component({
  selector: 'app-servers',
  templateUrl: './servers.component.html',
  styleUrls: ['./servers.component.css']
})
export class ServersComponent implements OnInit {
  public servers: {id: number, name: string, status: string}[] = [];

  constructor(private serversService: ServersService,
              private router: Router,
              private route: ActivatedRoute) { }

  ngOnInit() {
    this.servers = this.serversService.getServers();
  }
  onReload() {
    this.router.navigate(['servers'], {relativeTo: this.route})
  }
}
```

Relative paths for Navigating Programmatically

The screenshot shows a web browser window with the URL `http://localhost:4200/servers`. The page displays a navigation bar with tabs for Home, Servers, and Users. Under the Servers tab, there is a list of servers: Productionserver, Testserver, and Devserver. To the right of this list is a form with fields for Server Name (set to Productionserver) and Server Status (set to Online). A blue "Update Server" button is at the bottom of the form. Below the page content is the Chrome DevTools Console tab, which is currently active. The console shows the following error message:

```
core.js:27681 index.js:52 core.js:5967
ERROR Error: Uncaught (in promise): Error: Cannot match any routes. URL Segment: 'servers/servers'
Error: Cannot match any routes. URL Segment: 'servers/servers'
    at ApplyRedirects.noMatchError (apply_redirects.ts:117)
    at CatchSubscriber.selector (apply_redirects.ts:94)
    at CatchSubscriber.error (catchError.ts:130)
    at MapSubscriber._error (Subscriber.ts:143)
    at MapSubscriber.error (Subscriber.ts:113)
    at MapSubscriber._error (Subscriber.ts:143)
    at MapSubscriber.error (Subscriber.ts:113)
    at MapSubscriber._error (Subscriber.ts:143)
    at MapSubscriber.error (Subscriber.ts:113)
    at ThrowIfEmptySubscriber._error (Subscriber.ts:143)
    at resolvePromise (zone.js:832)
    at resolvePromise (zone.js:784)
    at zone.js:894
    at ZoneDelegate.invokeTask (zone.js:421)
    at Object.onInvokeTask (core.js:28269)
    at ZoneDelegate.invokeTask (zone.js:420)
    at Zone.runTask (zone.js:188)
    at drainMicroTaskQueue (zone.js:601)
    at ZoneTask.invokeTask [as invoke] (zone.js:507)
    at invokeTask (zone.js:1671)
```

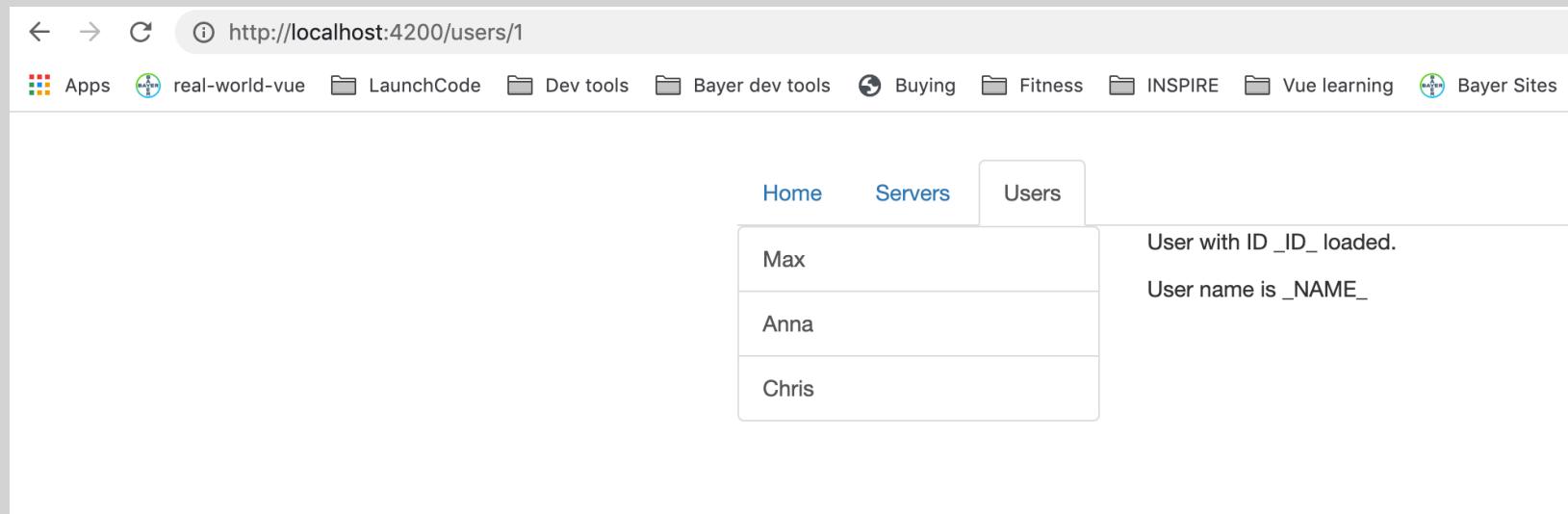
Passing Parameters to Routes

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

@Component({
  selector: 'app-users',
  templateUrl: './users.component.html',
  styleUrls: ['./users.component.css']
})
export class UsersComponent {
  users = [
    {
      id: 1,
      name: 'Max'
    },
    {
      id: 2,
      name: 'Anna'
    },
    {
      id: 3,
      name: 'Chris'
    }
  ];
}
```

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'user', component: UsersComponent},
  { path: 'users/:id', component: UsersComponent},
  { path: 'server', component: ServersComponent},
  { path: 'servers', component: ServersComponent}
];
```

Passing Parameters to Routes



Fetching Route Parameters

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'user', component: UsersComponent},
  { path: 'users/:id/:name', component: UsersComponent},
  { path: 'server', component: ServersComponent},
  { path: 'servers', component: ServersComponent}
];
app.module.ts
```

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

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {
  user: {id: number, name: string};

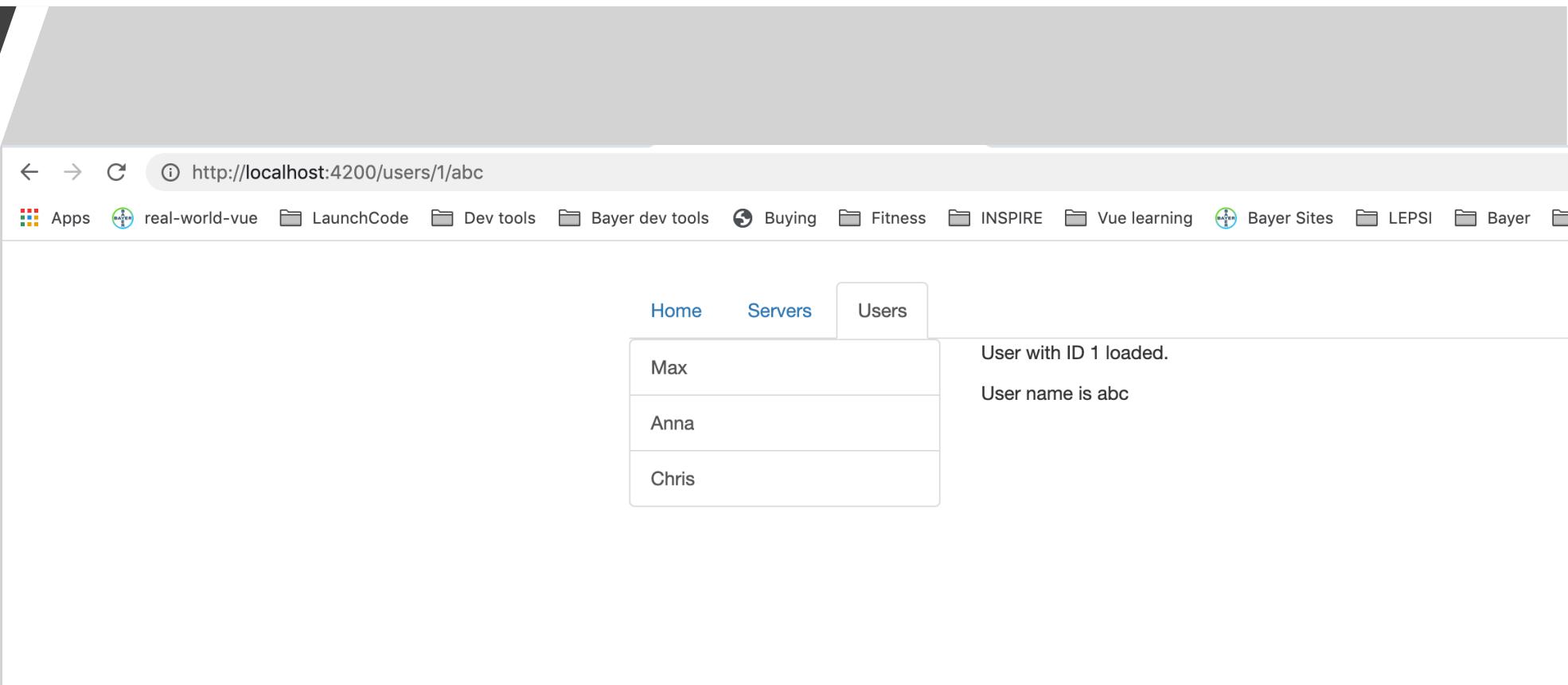
  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.user = {
      id: this.route.snapshot.params['id'],
      name: this.route.snapshot.params['name']
    }
  }
}
```

```
user.component.ts
```

```
<p>User with ID {{ user.id }} loaded.</p>
<p>User name is {{ user.name }}</p>
user.component.html
```

Testing route parameters



Fetching Route parameters reactively

this.route.params is an Observable

params might change in the future, if a user clicks, but u don't know when. Can't block code.

Observables are a feature added by a third party package, heavily used by Angular

Allow you to work with asynchronous tasks.

Observable is a way to subscribe to an event that might occur in the future to then execute that code when it happens.

Fetching Route parameters reactively

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

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {
  user: {id: number, name: string};

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.user = {
      id: this.route.snapshot.params['id'],
      name: this.route.snapshot.params['name']
    }
    this.route.params
      .subscribe(
        (params: Params) => {
          this.user.id = params['id'];
          this.user.name = params['name'];
        }
      );
  }
}
```

Fetching Route parameters reactively

A screenshot of a web browser window displaying a user profile page. The URL in the address bar is `http://localhost:4200/users/10/Anna`. The browser's toolbar includes back, forward, and refresh buttons. Below the address bar is a horizontal navigation bar with links: Apps, real-world-vue, LaunchCode, Dev tools, Bayer dev tools, Buying, Fitness, INSPIRE, Vue learning, and Bayer Si. The main content area shows a navigation menu with tabs: Home, Servers, and Users. The Users tab is active and highlighted with a blue border. Below the tabs is a list of users: Max, Anna, and Chris. To the right of the list, a message states "User with ID 10 loaded." and "User name is Anna". At the bottom right of the list, there is a link "Load Anna(10)".

Home Servers Users

Max

Anna

Chris

User with ID 10 loaded.

User name is Anna

Load Anna(10)

Route Observables

In order to destroy the route subscription, delete it in onDestroy method.

```
import { ActivatedRoute, Params } from '@angular/router';
import { Component, OnInit } from '@angular/core';
import { Subscription } from 'rxjs';

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {
  user: {id: number, name: string};
  paramsSubscription: Subscription

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.user = {
      id: this.route.snapshot.params['id'],
      name: this.route.snapshot.params['name']
    }
    this.paramsSubscription = this.route.params
      .subscribe(
        (params: Params) => {
          this.user.id = params['id'];
          this.user.name = params['name'];
        }
      );
  }
}
```

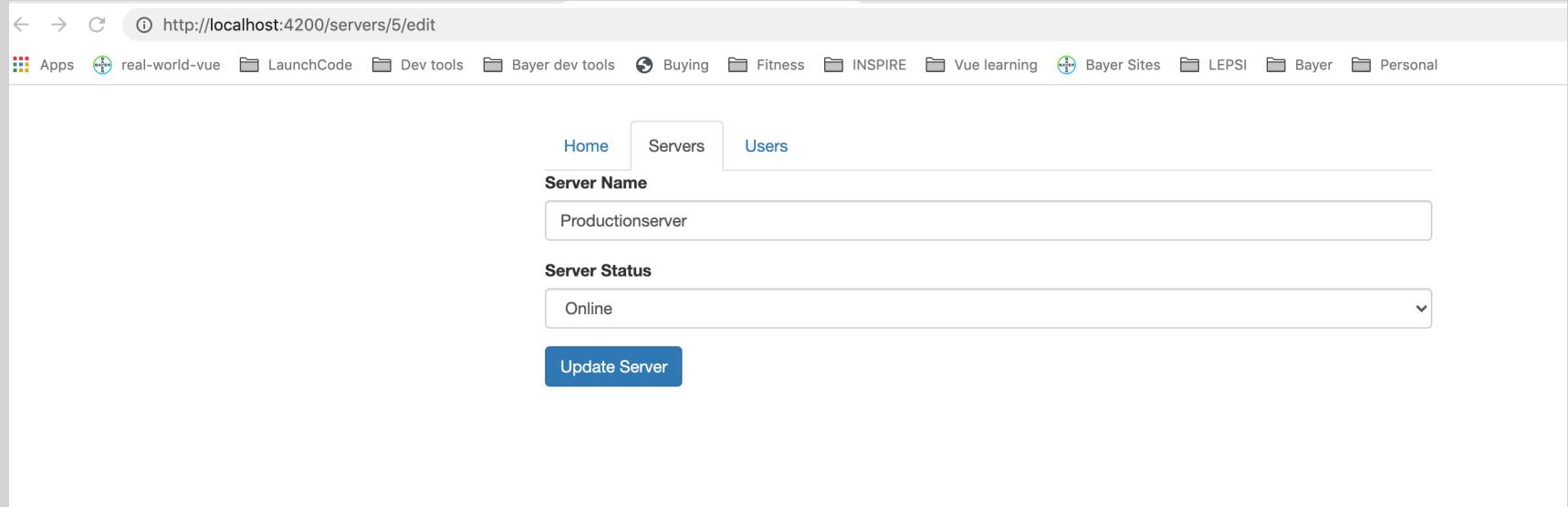
Passing Query parameters and fragments

Passing and retrieving
localhost:4200/users/10/Anna?mode=editing#loading

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'user', component: UsersComponent},
  { path: 'users/:id/:name', component: UsersComponent},
  { path: 'servers', component: ServersComponent},
  { path: 'servers/:id/edit', component: EditServerComponent}
];
```

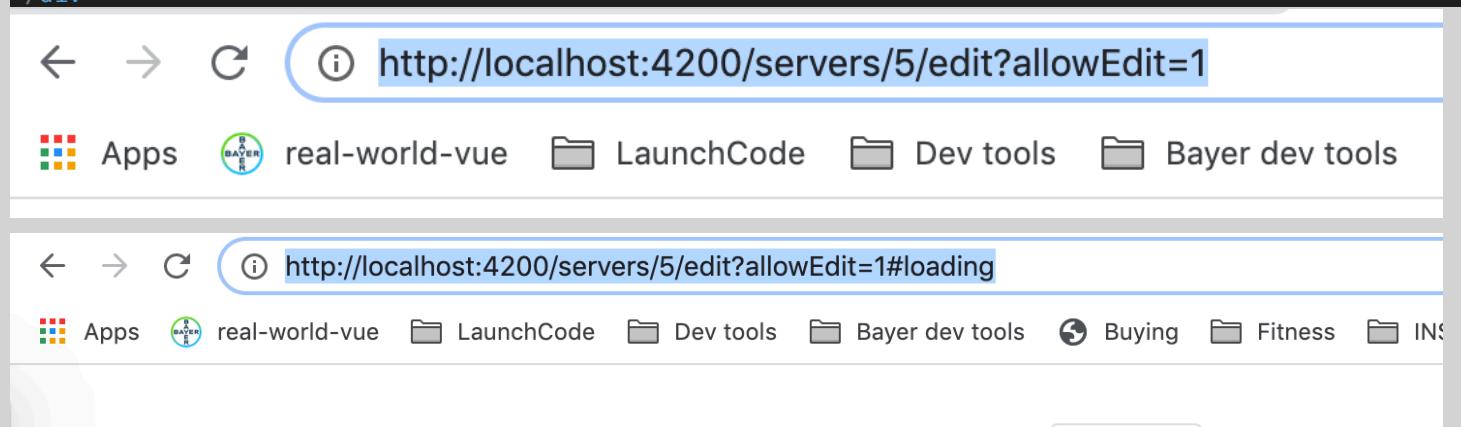
```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["'/servers', 5, 'edit']"
        href="#"
        class="list-group-item"
        *ngFor="let server of servers">
        {{ server.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr>
    <app-server></app-server>
  </div>
</div>
```

Passing Query parameters and fragments



Passing Query parameters and fragments

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["'/servers', 5, 'edit']"
        [queryParams]="{{allowEdit: '1'}}"
        fragment="loading"
        href="#"
        class="list-group-item"
        *ngFor="let server of servers">
        {{ server.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr>
    <app-server></app-server>
  </div>
</div>
```



Passing Query parameters and fragments

```
<h4>Welcome to Server Manager 4.0</h4>
<p>Manage your Servers and Users.</p>
<button class="btn btn-primary" (click)="onLoadServer(1)">Load Server 1</button>
```

home.component.html

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit() {
  }

  onLoadServer(id: number) {
    this.router.navigate(['/servers', id, 'edit'],
      {queryParams: {allowEdit: '1'},
       fragment: 'loading'});
  }
}
```

home.component.ts

Passing Query parameters and fragments

The screenshot shows a web browser window with the URL `http://localhost:4200` in the address bar. The page title is "Welcome to Server Manager 4.0". Below the title, there is a sub-header "Manage your Servers and Users." and a blue button labeled "Load Server 1". At the top of the page, there is a navigation bar with three tabs: "Home" (which is active), "Servers", and "Users". The top navigation bar also includes links for "Apps", "real-world-vue", "LaunchCode", "Dev tools", "Bayer dev tools", "Buying", "Fitness", "INSPIRE", and "Vue".

The screenshot shows a web browser window with the URL `http://localhost:4200/servers/1/edit?allowEdit=1#loading` in the address bar. The page displays a form for editing a server. The "Servers" tab is active in the navigation bar. The form fields include "Server Name" (set to "Productionserver") and "Server Status" (set to "Online"). There is also a "Update Server" button at the bottom of the form. The top navigation bar includes links for "Apps", "real-world-vue", "LaunchCode", "Dev tools", "Bayer dev tools", "Buying", "Fitness", "INSPIRE", "Vue learning", "Bayer Sites", "LEPSI", "Bayer", and "Personal".

Retrieving Query parameters and fragments

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

import { ServersService } from '../servers.service';

@Component({
  selector: 'app-edit-server',
  templateUrl: './edit-server.component.html',
  styleUrls: ['./edit-server.component.css']
})
export class EditServerComponent implements OnInit {
  server: {id: number, name: string, status: string};
  serverName = '';
  serverStatus = '';

  constructor(private serversService: ServersService,
    private route: ActivatedRoute) { }

  ngOnInit() {
    console.log(this.route.snapshot.queryParams);
    console.log(this.route.snapshot.fragment);
    // this.route.queryParams.subscribe();
    // this.route.fragment.subscribe();
    this.server = this.serversService.getServer(1);
    this.serverName = this.server.name;
    this.serverStatus = this.server.status;
  }

  onUpdateServer() {
    this.serversService.updateServer(this.server.id, {name: this.serverName, status: this.serverStatus});
  }
}
```

Retrieving Query parameters and fragments

The screenshot shows a web browser window with the URL `http://localhost:4200/servers/1/edit?allowEdit=1#loading`. The page displays a form for editing a server, with tabs for Home, Servers, and Users. The Servers tab is active, showing fields for Server Name (Productionserver) and Server Status (Online). A blue "Update Server" button is at the bottom. Below the browser window is an open developer tools console. The console has tabs for Elements, Sources, Console, Network, Performance, Memory, Security, Application, Lighthouse, and Augury. The Console tab is selected. The output area shows:

```
x Expression  
not available  
Angular is running in development mode. Call enableProdMode() to enable production mode.  
[WDS] Live Reloading enabled.  
allowEdit: "1"  
allowEdit: "1"  
__proto__: Object  
constructor: f hasOwnProperty()  
hasOwnProperty: f isPrototypeOf()  
isPrototypeOf: f propertyIsEnumerable()  
propertyIsEnumerable: f toLocaleString()  
toLocaleString: f toString()  
toString: f valueOf()  
valueOf: f  
__defineGetter__: f __defineGetter__()  
__defineSetter__: f __defineSetter__()  
__lookupGetter__: f __lookupGetter__()  
__lookupSetter__: f __lookupSetter__()  
get __proto__: f __proto__()  
set __proto__: f __proto__()  
loading
```

Annotations indicate file paths for some of the messages: `core.js:27681`, `index.js:52`, `edit-server.component.ts:20`, and `edit-server.component.ts:21`.

Cleaning up

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["/users", user.id, user.name]"
        href="#"
        class="list-group-item"
        *ngFor="let user of users">
        {{ user.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <app-user></app-user>
  </div>
</div>
```

users.component.html

Cleaning up

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["/servers", server.id]
        [queryParams]={allowEdit: '1'}
        fragment="loading"
        href="#"
        class="list-group-item"
        *ngFor="let server of servers">
        {{ server.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr>
    <app-server></app-server>
  </div>
</div>
```

servers.component.html

Cleaning up

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'user', component: UsersComponent},
  { path: 'users/:id/:name', component: UsersComponent},
  { path: 'servers', component: ServersComponent},
  { path: 'servers/:id', component: ServerComponent},
  { path: 'servers/:id/edit', component: EditServerComponent}
];
```

app.module.ts

Cleaning up

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Params, Router } from '@angular/router';

import { ServersService } from '../servers.service';

@Component({
  selector: 'app-server',
  templateUrl: './server.component.html',
  styleUrls: ['./server.component.css']
})
export class ServerComponent implements OnInit {
  server: {id: number, name: string, status: string};

  constructor(private serversService: ServersService,
    private route: ActivatedRoute) { }

  ngOnInit() {
    this.server = this.serversService.getServer(this.route.snapshot.params['id']);
    this.route.params.subscribe((params: Params) => {
      this.server = this.serversService.getServer(params['id']);
    })
  }
}
```

server.component.ts

Cleaning up

The screenshot shows a web application interface for managing servers. At the top, there's a navigation bar with links like 'Home', 'Servers' (which is active), and 'Users'. Below this is a sidebar listing 'Productionserver', 'Testserver', and 'Devserver'. To the right, there's a form with fields for 'Server Name' (set to 'Productionserver') and 'Server Status' (set to 'Online'), along with a 'Reload Page' button and a large blue 'Update Server' button.

At the bottom, a browser's developer tools console is open. The 'Console' tab is selected. The output area shows several messages:

```
: Expression not available
Angular is running in development mode. Call enableProdMode() to enable production mode.
▶ {followEdit: "I"}
loading
[WDS] Live Reloading enabled.
▶ {}
undefined
2 ▶ ERROR TypeError: Cannot read property 'name' of undefined
    at ServerComponent_Template (template.html:1)
    at executeTemplate (core.js:9310)
    at refreshView (core.js:9179)
    at refreshComponent (core.js:10345)
    at refreshChildComponents (core.js:8976)
    at refreshView (core.js:9229)
    at refreshComponent (core.js:10345)
    at refreshChildComponents (core.js:8976)
    at refreshView (core.js:9229)
    at refreshEmbeddedViews (core.js:10299)
```

The error message at the bottom is highlighted with a red background, indicating a runtime error in the application code.

Cleaning up

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["/servers", server.id]
        [queryParams]={allowEdit: '1'}
        fragment="loading"
        href="#"
        class="list-group-item"
        *ngFor="let server of servers">
        {{ server.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr>
    <!-- <app-server></app-server> -->
  </div>
</div>
```

Cleaning up

A screenshot of a web browser window displaying a server management application at <http://localhost:4200/servers>. The browser's address bar shows the URL. The page has a header with navigation links: Home, Servers (which is active), and Users. Below this is a list of servers: Productionserver, Testserver, and Devserver. To the right of the server list is a form for updating a selected server. The form includes a "Reload Page" button, a "Server Name" field containing "Productionserver", a "Server Status" dropdown menu set to "Online", and a large blue "Update Server" button.

← → ⌂ ⓘ http://localhost:4200/servers

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Home Servers Users

Productionserver

Testserver

Devserver

Reload Page

Server Name

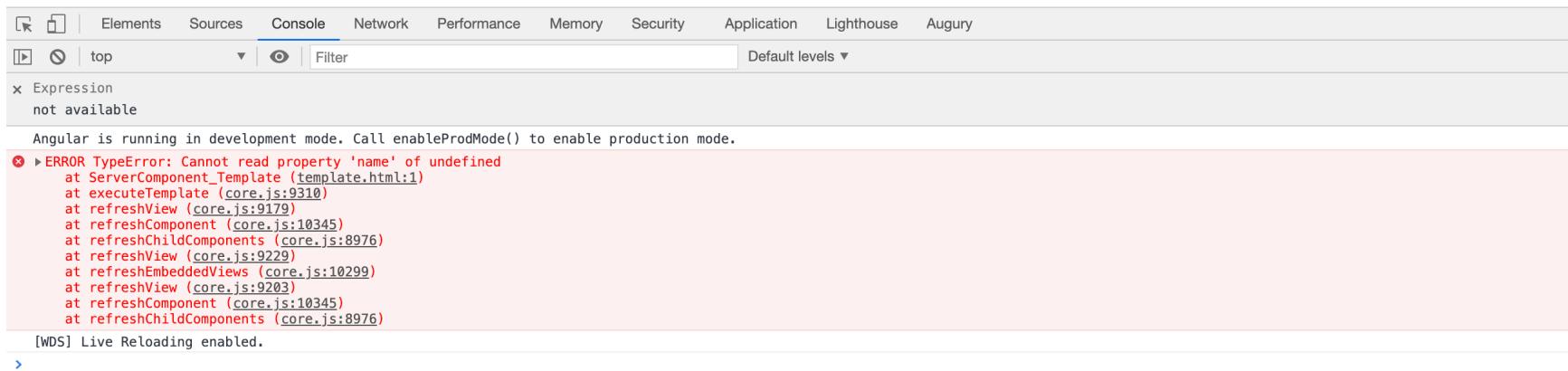
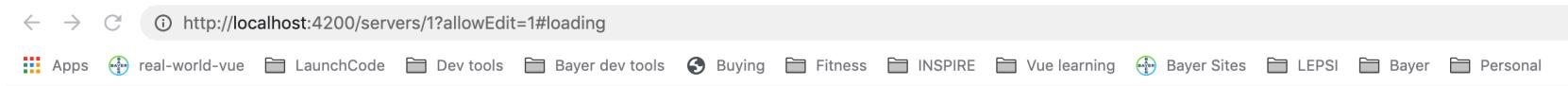
Productionserver

Server Status

Online

Update Server

Cleaning up



Cleaning up

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Params, Router } from '@angular/router';

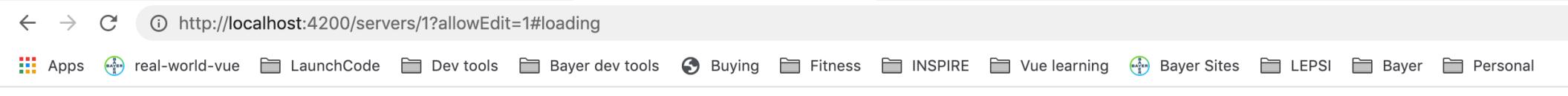
import { ServersService } from '../servers.service';

@Component({
  selector: 'app-server',
  templateUrl: './server.component.html',
  styleUrls: ['./server.component.css']
})
export class ServerComponent implements OnInit {
  server: {id: number, name: string, status: string};

  constructor(private serversService: ServersService,
    private route: ActivatedRoute) { }

  ngOnInit() {
    this.server = this.serversService.getServer(+this.route.snapshot.params['id']);
    this.route.params.subscribe((params: Params) => {
      this.server = this.serversService.getServer(+params['id']);
    })
  }
}
```

Cleaning up



Home Servers Users

Productionserver

Server status is online

Nested Routes

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'user', component: UsersComponent},
  { path: 'users/:id/:name', component: UsersComponent},
  { path: 'servers', component: ServersComponent},
  { path: 'servers/:id', component: ServerComponent},
  { path: 'servers/:id/edit', component: EditServerComponent}
];
```

Can add child routes to nest

Nested Routes

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'user', component: UsersComponent},
  { path: 'users/:id/:name', component: UsersComponent},
  { path: 'servers', component: ServersComponent,
    children: [{ path: ':id', component: ServerComponent},
      { path: ':id/edit', component: EditServerComponent}]},
];

```

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["/servers", server.id]
        [queryParams]="{{allowEdit: '1'}}"
        fragment="loading"
        href="#"
        class="list-group-item"
        *ngFor="let server of servers">
        {{ server.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <router-outlet></router-outlet>
    <!-- <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr> -->
    <!-- <app-server></app-server> -->
  </div>
</div>
```

Servers.component.html

Nested Routes



The screenshot shows a component structure for a "Servers" page. The main template contains a navigation bar with tabs: "Home", "Servers", and "Users". The "Servers" tab is active. Below the tabs, there is a list of server names: "Productionserver", "Testserver", and "Devserver". To the right of the list, under the "Servers" tab, there is a section titled "Devserver" with the subtext "Server status is offline". This illustrates how multiple components can be nested within a single route path.

Nested Routes

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'users', component: UsersComponent,
    children: [{ path: ':id/:name', component: UserComponent}]},
  { path: 'servers', component: ServersComponent,
    children: [{ path: ':id', component: ServerComponent},
      { path: ':id/edit', component: EditServerComponent}]},
];

```

```
<p>User with ID {{ user.id }} loaded.</p>
<p>User name is {{ user.name }}</p>
<hr>
<a [routerLink]=["/users", 10, 'Anna']>Load Anna(10)</a>
```

user.component.html

Nested Routes

← → ⌂ ⓘ http://localhost:4200/users/3/Chris

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Home Servers Users

Max
Anna
Chris

User with ID 3 loaded.
User name is Chris

[Load Anna\(10\)](#)
User with ID loaded.
User name is

[Load Anna\(10\)](#)

Using Query Parameters

```
<h5>{{ server.name }}</h5>
<p>Server status is {{ server.status }}</p>
<button class="btn btn-primary" (click)="onEdit()">Edit Server</button>
```

server.component.html

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Params, Router } from '@angular/router';

import { ServersService } from '../servers.service';

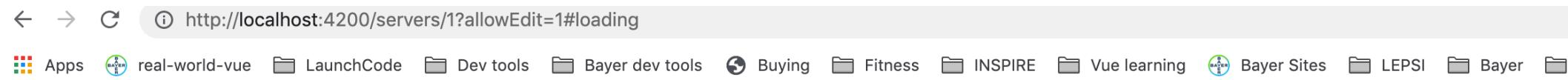
@Component({
  selector: 'app-server',
  templateUrl: './server.component.html',
  styleUrls: ['./server.component.css']
})
export class ServerComponent implements OnInit {
  server: {id: number, name: string, status: string};

  constructor(private serversService: ServersService,
    private route: ActivatedRoute,
    private router: Router) { }

  ngOnInit() {
    this.server = this.serversService.getServer(+this.route.snapshot.params['id']);
    this.route.params.subscribe((params: Params) => {
      this.server = this.serversService.getServer(+params['id']);
    })
  }
  onEdit() {
    this.router.navigate(['edit'], {relativeTo: this.route})
  }
}
```

server.component.ts

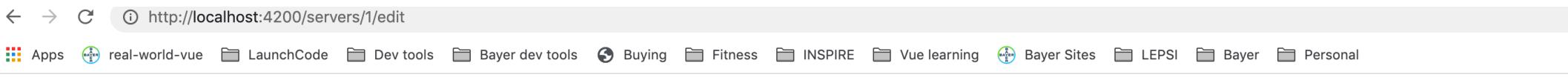
Using Query Parameters



Home Servers Users

Productionserver
Testserver
Devserver

Productionserver
Server status is online
Edit Server



Home Servers Users

Productionserver
Testserver
Devserver

Server Name

Server Status

Update Server

Using Query Parameters

```
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="list-group">
      <a
        [routerLink]=["/servers", server.id]
        [queryParams]={allowEdit: server.id === 3 ? '1' : '0'}
        fragment="loading"
        href="#"
        class="list-group-item"
        *ngFor="let server of servers">
        {{ server.name }}
      </a>
    </div>
  </div>
  <div class="col-xs-12 col-sm-4">
    <router-outlet></router-outlet>
    <!-- <button class="btn btn-primary" (click)="onReload()">Reload Page</button>
    <app-edit-server></app-edit-server>
    <hr> -->
    <!-- <app-server></app-server> -->
  </div>
</div>
```

servers.component.html

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

import { ServersService } from '../servers.service';

@Component({
  selector: 'app-edit-server',
  templateUrl: './edit-server.component.html',
  styleUrls: ['./edit-server.component.css']
})
export class EditServerComponent implements OnInit {
  server: {id: number, name: string, status: string};
  serverName = '';
  serverStatus = '';
  allowEdit = false;

  constructor(private serversService: ServersService,
    private route: ActivatedRoute) { }

  ngOnInit() {
    console.log(this.route.snapshot.queryParams);
    console.log(this.route.snapshot.fragment);
    this.route.queryParams.subscribe(
      (queryParams: Params) => {
        this.allowEdit = queryParams['allowEdit'] === '1' ? true: false
      }
    );
    this.route.fragment.subscribe();
    this.server = this.serversService.getServer(1);
    this.serverName = this.server.name;
    this.serverStatus = this.server.status;
  }

  onUpdateServer() {
    this.serversService.updateServer(this.server.id, {name: this.serverName, status: this.serverStatus});
  }
}
```

edit-server.component.ts

Using Query Parameters

```
<h4 *ngIf="!allowEdit">You are not allowed to edit!</h4>
<div *ngIf="allowEdit">
  <div class="form-group">
    <label for="name">Server Name</label>
    <input
      type="text"
      id="name"
      class="form-control"
      [(ngModel)]="serverName">
  </div>
  <div class="form-group">
    <label for="status">Server Status</label>
    <select
      id="status"
      class="form-control"
      [(ngModel)]="serverStatus">
      <option value="online">Online</option>
      <option value="offline">Offline</option>
    </select>
  </div>
  <button
    class="btn btn-primary"
    (click)="onUpdateServer()">Update Server</button>
</div>
```

edit-server.component.html

Using Query Parameters

A screenshot of a web browser window. The address bar shows the URL `http://localhost:4200/servers/1/edit`. The page content is a 403 Forbidden error. At the top, there is a navigation bar with links: Home, Servers (which is the active tab), and Users. Below this is a list of three servers: Productionserver, Testserver, and Devserver. To the right of the servers, the message **You are not allowed to edit!** is displayed.

← → ⌂ ⓘ http://localhost:4200/servers/1/edit

Apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites LEPSI Bayer Personal

Home Servers Users

Productionserver

Testserver

Devserver

You are not allowed to edit!

Using Query Parameters

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Params, Router } from '@angular/router';

import { ServersService } from '../servers.service';

@Component({
  selector: 'app-server',
  templateUrl: './server.component.html',
  styleUrls: ['./server.component.css']
})
export class ServerComponent implements OnInit {
  server: {id: number, name: string, status: string};

  constructor(private serversService: ServersService,
    private route: ActivatedRoute,
    private router: Router) { }

  ngOnInit() {
    this.server = this.serversService.getServer(+this.route.snapshot.params['id']);
    this.route.params.subscribe((params: Params) => {
      this.server = this.serversService.getServer(+params['id']);
    })
  }
  onEdit() {
    this.router.navigate(['edit'], {relativeTo: this.route, queryParamsHandling: 'preserve'})
  }
}
```

Using Query Parameters

← → ⌂ ⓘ http://localhost:4200/servers/3?allowEdit=1#loading

_apps real-world-vue LaunchCode Dev tools Bayer dev tools Buying Fitness INSPIRE Vue learning Bayer Sites

Home Servers Users

Productionserver	Devserver Server status is offline
Testserver	
Devserver	Edit Server

Redirecting and Wildcard Routes

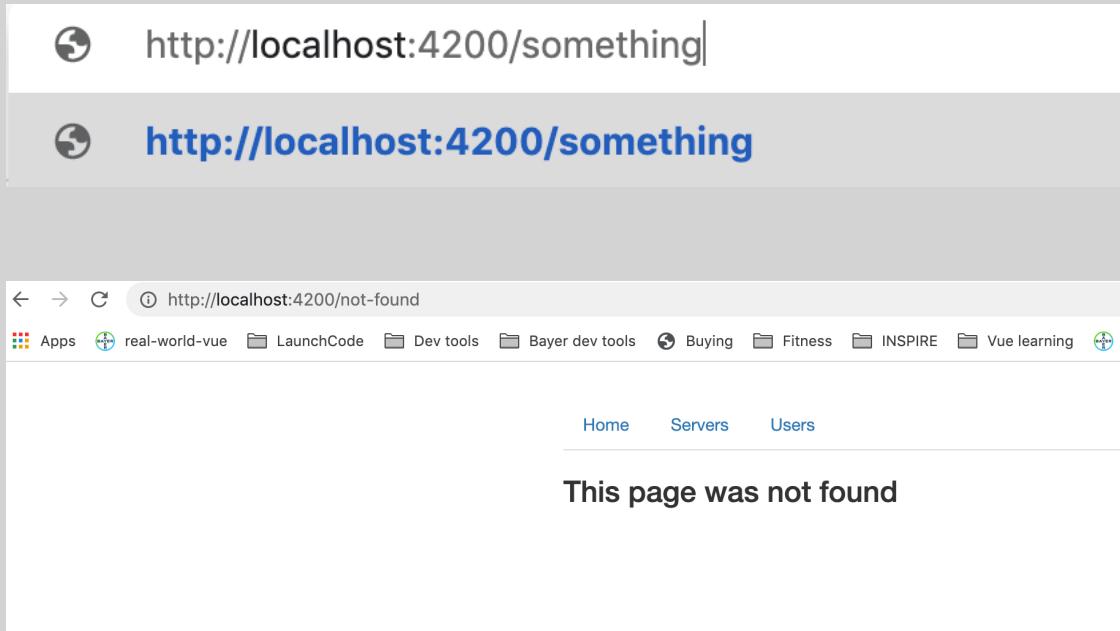
- ng g c page-not-found

```
<h3>This page was not found</h3>
```

page-not-found.component.html

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'users', component: UsersComponent,
    children: [{ path: ':id/:name', component: UserComponent}]},
  { path: 'servers', component: ServersComponent,
    children: [{ path: ':id', component: ServerComponent},
      { path: ':id/edit', component: EditServerComponent}]},
  { path: 'not-found', component: PageNotFoundComponent},
  { path: 'something', redirectTo: '/not-found'}
];
```

Redirecting and Wildcard Routes



Redirecting and Wildcard Routes

```
const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'users', component: UsersComponent,
    children: [{ path: ':id/:name', component: UserComponent}]},
  { path: 'servers', component: ServersComponent,
    children: [{ path: ':id', component: ServerComponent},
      { path: ':id/edit', component: EditServerComponent}]},
  { path: 'not-found', component: PageNotFoundComponent},
  { path: '**', redirectTo: '/not-found'}
];
```

Catch all paths you don't know. The generic route should be at the bottom otherwise all paths take you there!

Outsourcing the Route Configuration

- `app-routing.module.ts`

```
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { HomeComponent } from './home/home.component';
import { NgModule } from "@angular/core";
import { Routes, RouterModule } from "@angular/router";
import { UsersComponent } from './users/users.component';
import { ServersComponent } from './servers/servers.component';
import { EditServerComponent } from './servers/edit-server/edit-server.component';
import { UserComponent } from './users/user/user.component';

const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'users', component: UsersComponent,
    children: [{ path: ':id/:name', component: UserComponent}]},
  { path: 'servers', component: ServersComponent,
    children: [{ path: ':id', component: ServersComponent},
      { path: ':id/edit', component: EditServerComponent}]},
  { path: 'not-found', component: PageNotFoundComponent},
  { path: '**', redirectTo: '/not-found'}
];

@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
  ],
  exports: [RouterModule]
})
export class AppRoutingModule {
```

Outsourcing the Route Configuration

- **app.module.ts**

```
import { AppRoutingModule } from './app.routing.module';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { UsersComponent } from './users/users.component';
import { ServersComponent } from './servers/servers.component';
import { UserComponent } from './users/user/user.component';
import { EditServerComponent } from './servers/edit-server/edit-server.component';
import { ServerComponent } from './servers/server/server.component';
import { ServersService } from './servers/servers.service';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    UsersComponent,
    ServersComponent,
    UserComponent,
    EditServerComponent,
    ServerComponent,
    PageNotFoundComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ]
})
```

Guards

- Route Guards – functionality before/after entering a route
- canActivate guard - auth-guard.service.ts

```
import { Injectable } from "@angular/core";
import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from "@angular/router";
import { Observable } from "rxjs";
import { AuthService } from "./auth.service";

@Injectable()
export class AuthGuard implements CanActivate{

    constructor(private authService: AuthService, private router: Router) {}

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
        return this.authService.isAuthenticated()
            .then(
                (authenticated: boolean) => {
                    if (authenticated) {
                        return true;
                    } else {
                        this.router.navigate(['/']);
                    }
                }
            );
    }
}
```

Guards

- auth.service.ts

```
export class AuthService {  
    loggedIn = false;  
  
    isAuthenticated() {  
        const promise = new Promise(  
            (resolve, reject) => {  
                setTimeout(() => {  
                    resolve(this.loggedIn);  
                }, 800);  
            }  
        );  
        return promise;  
    }  
    login() {  
        this.loggedIn = true;  
    }  
    logout() {  
        this.loggedIn = false;  
    }  
}
```

Guards

- **app.module.ts**

```
import { AuthService } from './auth.service';
import { AppRoutingModule } from './app.routing.module';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { UsersComponent } from './users/users.component';
import { ServersComponent } from './servers/servers.component';
import { UserComponent } from './users/user/user.component';
import { EditServerComponent } from './servers/edit-server/edit-server.component';
import { ServerComponent } from './servers/server/server.component';
import { ServersService } from './servers/servers.service';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { AuthGuard } from './auth-guard.service';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    UsersComponent,
    ServersComponent,
    UserComponent,
    EditServerComponent,
    ServerComponent,
    PageNotFoundComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: [
    AuthService,
    AuthGuard
  ]
})
```

Guards

- app.routing.module.ts

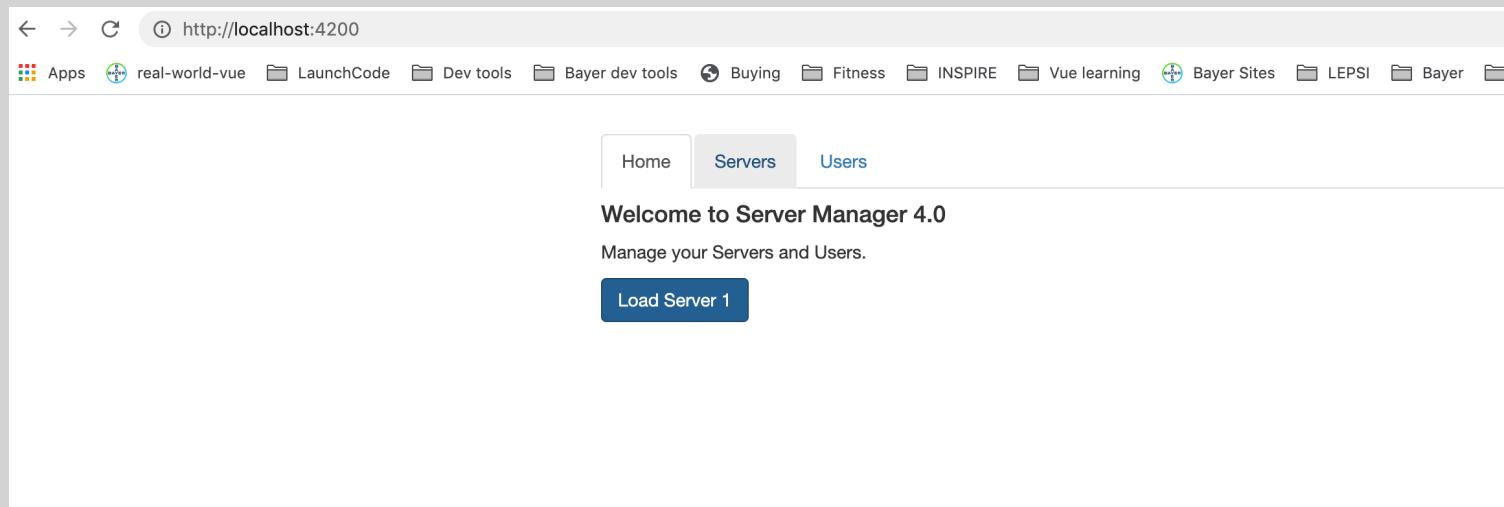
```
import { AuthGuard } from './auth-guard.service';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { HomeComponent } from './home/home.component';
import { NgModule } from "@angular/core";
import { Routes, RouterModule } from "@angular/router";
import { UsersComponent } from './users/users.component';
import { ServersComponent } from './servers/servers.component';
import { EditServerComponent } from './servers/edit-server/edit-server.component';
import { UserComponent } from './users/user/user.component';

const appRoutes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'users', component: UsersComponent,
    children: [{ path: ':id/:name', component: UserComponent }],
  },
  { path: 'servers', canActivate: [AuthGuard], component: ServersComponent,
    children: [{ path: ':id', component: ServersComponent },
      { path: ':id/edit', component: EditServerComponent }]
  },
  { path: 'not-found', component: PageNotFoundComponent },
  { path: '**', redirectTo: '/not-found' }
];

@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
  ],
  exports: [RouterModule]
})
export class AppRoutingModule {
```

Guards

- Servers does not take you anywhere at first and after 800 sec it takes you.



Protecting Child Routes

- Can specify canActivate to each child
- There is also canActivateChild interface/

```
import { AuthService } from './auth.service';
import { AuthGuard } from './auth-guard.service';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { HomeComponent } from './home/home.component';
import { NgModule } from "@angular/core";
import { Routes, RouterModule } from "@angular/router";
import { UsersComponent } from './users/users.component';
import { ServersComponent } from './servers/servers.component';
import { EditServerComponent } from './servers/edit-server/edit-server.component';
import { UserComponent } from './users/user/user.component';

const appRoutes: Routes = [
  { path: '', component: HomeComponent},
  { path: 'users', component: UsersComponent,
    children: [{ path: ':id/:name', component: UserComponent}],
    {
      path: 'servers',
      // canActivate: [AuthGuard],
      canActivateChild: [AuthGuard],
      component: ServersComponent,
      children: [{ path: ':id', component: ServersComponent},
        { path: ':id/edit', component: EditServerComponent}],
      { path: 'not-found', component: PageNotFoundComponent},
      { path: '**', redirectTo: '/not-found'}
    ]
  },
  @NgModule({
    imports: [
      RouterModule.forRoot(appRoutes)
    ],
    exports: [RouterModule]
  })
export class AppRoutingModule { }
```

app.routing.module.ts

Protecting Child Routes

auth-guard.service.ts

```
import { Injectable } from "@angular/core";
import { ActivatedRouteSnapshot, CanActivate, CanActivateChild, Router, RouterStateSnapshot, UrlTree } from
"@angular/router";
import { Observable } from "rxjs";
import { AuthService } from "./auth.service";

@Injectable()
export class AuthGuard implements CanActivate, CanActivateChild{

    constructor(private authService: AuthService, private router: Router) {}

    canActivateChild(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree |
Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
        return this.canActivate(route, state);
    }

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree |
Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {
        return this.authService.isAuthenticated()
            .then(
                (authenticated: boolean) => {
                    if (authenticated) {
                        return true;
                    } else {
                        this.router.navigate(['/']);
                    }
                }
            );
    }
}
```

Using Fake Auth Service

```
<h4>Welcome to Server Manager 4.0</h4>
<p>Manage your Servers and Users.</p>
<button class="btn btn-primary" (click)="onLoadServer(1)">Load Server 1</button>
<button class="btn btn-default" (click)="onLogin()">Login</button>
<button class="btn btn-default" (click)="onLogout()">Logout</button>
```

home.component.html

```
import { AuthService } from './auth.service';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor(private router: Router, private authService: AuthService) { }

  ngOnInit() {
  }

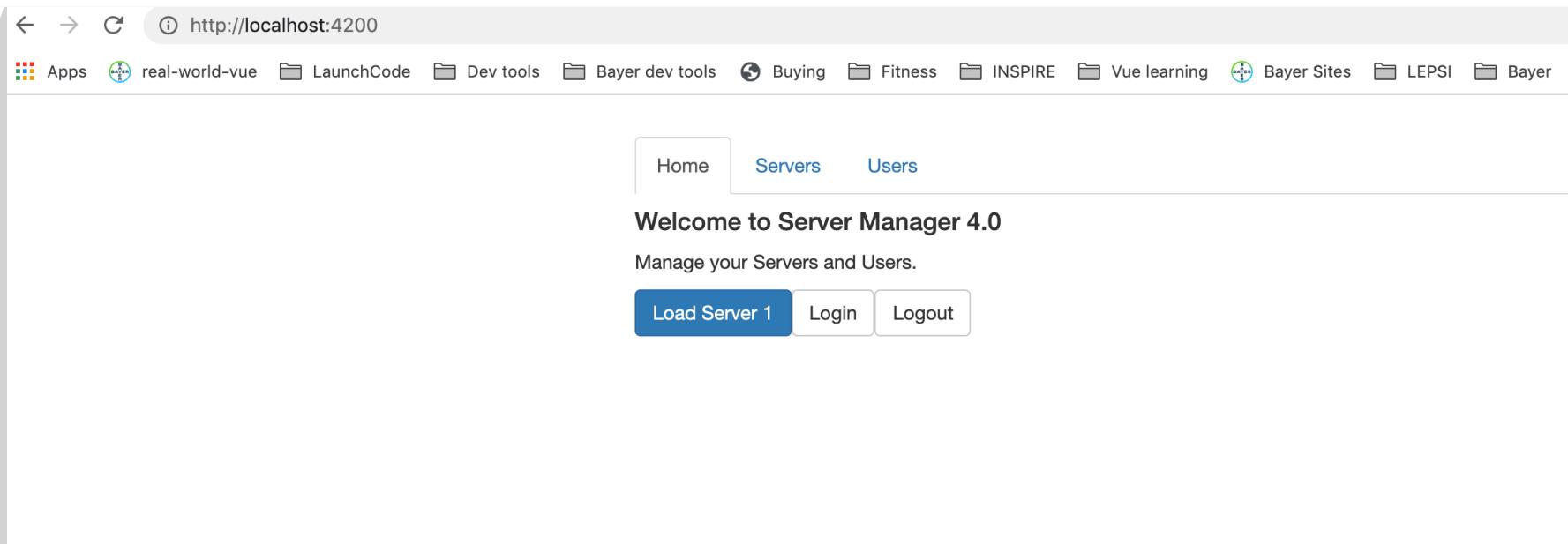
  onLoadServer(id: number) {
    this.router.navigate(['/servers', id, 'edit'],
    {queryParams: {allowEdit: '1'}, fragment: 'loading'});
  }

  onLogin() {
    this.authService.login();
  }

  onLogout() {
    this.authService.logout();
  }
}
```

home.component.ts

Using Fake Auth Service



If you logout then it takes you to Home tab and login works

canDeactivate

Convenience method to prevent user from accidentally navigating away.

Summary

Questions?

Studio

Assignment 5: Practicing Services

Section 10: Course Project - Services & Dependency Injection