



R a i s i n g   t h e   b a r

# Angular Service & Router

# Mục tiêu

- Tạo được service tùy biến.
- Nhúng được service vào trong component.
- Triển khai được cơ chế router.

# Project mẫu

- Project mẫu tại folder: angular-service-router
- Hoặc bạn có thể tạo mới project với câu lệnh:

```
ng new angular-service-router --style scss
```

# Project mẫu

- Nếu bạn tạo mới project, bạn sẽ cần cài đặt thêm thư viện sau.
  - Các bạn chạy lệnh **npm i date-fns -S** để cài đặt thư viện cho việc tính toán datetime.
  - Các bạn chạy lệnh **npm i typescript@2.9.2 -D** để cài đặt thư viện typescript **nếu phiên bản hiện tại < 2.9.2**

# Angular service

# Chuẩn bị

ng generate component dictionary

ng generate service dictionary

# Angular service là gì?

- Service là những phần code được sử dụng lại giữa nhiều component với nhau.
- Ví dụ như khi bạn làm việc với giao dịch viên ở ngân hàng. Những giao dịch viên này có thể phục vụ nhiều khách hàng với nhiều mục đích khác nhau, và bạn không cần quan tâm hoạt động phía dưới như thế nào, chỉ cần quan tâm công việc của bạn có hoàn thành hay không (chuyển tiền thành công chặng hạn). Lúc này những giao dịch viên kia hiểu về nghiệp vụ (business), có thể phục vụ hết khách hàng này đến khách hàng khác, mà khách hàng không cần làm từ đầu đến cuối, khách hàng ủy thác việc đó cho họ. Nên họ có thể coi như Service, còn khách hàng là Component/Service khác.

# Angular service

- Angular distinguishes components from services to increase modularity and reusability.
- By separating a component's view-related functionality from other kinds of processing, you can make your component classes lean and efficient.

# Angular service

- Limit logic in a component to only that required for the view. All other logic should be delegated to services.
- Move reusable logic to services and keep components simple and focused on their intended purpose.

# Angular service

- Logic may be reused by multiple components when placed within a service and exposed via a function.
- Logic in a service can more easily be isolated in a unit test, while the calling logic in the component can be easily mocked.
- Removes dependencies and hides implementation details from the component.
- Keeps the component slim, trim, and focused.

# Tạo service – dictionary.service.ts

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

```
@Injectable({  
  providedIn: 'root'  
})
```

```
export class DictionaryService {  
}
```

Khai báo class phía dưới là một service, và nó  
được đặt trong scope của root, như thế sẽ  
tồn tại service này trong cả app

# Tạo service – dictionary.service.ts

```
export interface IWord {  
    key: string;  
    meaning: string;  
}
```

# Tạo service – dictionary.service.ts

```
export class DictionaryService {  
    private words: IWord[] = [  
        {key: 'service', meaning: 'dịch vụ'},  
        {key: 'service provider', meaning: 'nhà cung cấp dịch  
vụ'},  
        {key: 'provide', meaning: 'cung cấp'},  
        {key: 'efficient', meaning: 'hiệu quả'},  
        {key: 'snow', meaning: 'tuyết'}  
    ];  
}
```

# Tạo service – dictionary.service.ts

```
export class DictionaryService {
  search(word: string): string {
    if (!word) {
      return '';
    }
    const w = this.words.find(item => item.key === word.toLowerCase());
    if (w) {
      return w.meaning;
    }
    return 'Not found';
  }
}
```

# Đăng ký service

- Muốn sử dụng service bạn cần phải đăng ký nó.
- Có thể đăng ký ở cấp độ Component hoặc NgModule.
- Thông thường nếu bạn cần 1 instance duy nhất cho toàn bộ app thì hầu hết sẽ đăng ký ở root module như sau:

```
@Injectable({  
  providedIn: 'root'  
})
```

# Đăng ký service

- Đăng ký ở một NgModule cụ thể thì các thành phần trong NgModule đó sẽ sử dụng cùng một instance.

```
@Injectable()  
export class DictionaryService {}  
  
@NgModule({  
  providers: [DictionaryService],  
})  
export class AppModule { }
```

# Sử dụng service ở component

```
export class DictionaryComponent {  
    word: IWord;  
    constructor(private dictionaryService: DictionaryService) { }  
  
    search(word: string) {  
        const meaning = this.dictionaryService.search(word);  
        this.word = {  
            key: word,  
            meaning: meaning  
        };  
    }  
}
```

Component gửi yêu cầu đến Angular, Angular sẽ khởi tạo instance của service và nạp vào cho component thông qua constructor

Delegate sang phần xử lý của service

# Sử dụng service ở component

```
<h2>English-Vietnamese Dictionary</h2>
```

```
<input type="text"  
(change)="search($event.target.value)">
```

```
<p *ngIf="word">  
    Nghĩa của từ {{word.key}}: {{word.meaning}}  
</p>
```

# Sử dụng service ở component

## English-Vietnamese Dictionary

efficient

Nghĩa của từ efficient: hiệu quả

# Angular router

# Angular router

- Router cho phép ứng dụng có thể chia thành các pages, và có thể di chuyển từ page này sang page khác

# Sử dụng router trong ứng dụng

- Tạo mới file app-routing.module.ts trong thư mục src/app
- Khởi tạo mới một component: ng generate component dictionary-page
- Sau đó cài đặt các phần cần thiết cho router

# Sử dụng router trong ứng dụng

```
const routes: Routes = [ {  
    path: 'dictionary',  
    component: DictionaryPageComponent  
}];
```

# Sử dụng router trong ứng dụng

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

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

Do là root module, nên chúng ta sử dụng static method forRoot để đưa vào config về route, sau đó exports để AppModule sử dụng

# Sử dụng router trong ứng dụng

```
@NgModule({  
    imports: [  
        // other import  
        AppRoutingModule  
    ],  
})  
export class AppModule { }
```

# Config router-outlet

- app.component.html

```
<nav>
  <a routerLink="dictionary">Dictionary</a>
</nav>
```

Sử dụng directive này để Angular chặn việc chuyển trang thông thường, thay vào đó sẽ sử dụng client routing để có thể đưa người dùng đến page mong muốn

```
<router-outlet></router-outlet>
```

Điểm đánh dấu để Angular router có thể đặt component cần render sau khi người dùng navigate vào page.  
Directive này cần được đặt ở parent component, trong trường hợp này là app component

# Dictionary page

```
export class DictionaryPageComponent implements OnInit {  
  listWord: IWord[] = [];  
  constructor(  
    private dictionaryService: DictionaryService  
  ) {}  
  
  ngOnInit() {  
    this.listWord = this.dictionaryService.getAll();  
  }  
}
```

# Dictionary page

```
<h2>
  Dictionary page
</h2>

<ul>
  <li *ngFor="let word of listWord">
    {{ word.key }}
  </li>
</ul>
```

# Dictionary service

```
export class DictionaryService {  
    getAll(): IWord[] {  
        return this.words;  
    }  
}
```

# Angular router

- Từ home page, người dùng click vào link Dictionary, và bạn có thể thấy page dạng như sau

Dictionary

## Dictionary page

- service
- service provider
- provide
- efficient
- snow

# Angular router

- Angular router cung cấp service để bạn có thể chuyển trang từ trong component.
- Ví dụ: chúng ta có 1 button, khi người dùng click vào button đó thì di chuyển sang page tương ứng.

# Angular router

```
<button class="link" (click)="gotoTimelines()">Ứng  
dụng tính thời gian</button>
```

```
constructor(private router: Router) {}  
gotoTimelines() {  
  this.router.navigate(['/timelines']);  
}
```

# Child route

- Một route có thể chứa trong nó nhiều route con.
- Ví dụ với route dictionary vừa rồi, chúng ta có thể di chuyển vào từng word để xem thông tin chi tiết.
- Cú pháp config child route trong Angular như sau:

# Child route

```
const routes: Routes = [
{
  path: 'dictionary',
  component: DictionaryPageComponent,
  children: [
    {
      path: ':key',
      component: DictionaryDetailComponent
    }
  ]
};
```

Phần này sẽ là con của dictionary, và nó sẽ có thể chứa một giá trị dynamic như là một biến.  
Tên của nó được gọi là **param**.

# Child route

```
<ul>
  <li *ngFor="let word of listWord">
    <a [routerLink]=["/dictionary", word.key]">
      {{ word.key }}
    </a>
  </li>
</ul>

<router-outlet></router-outlet>
```

# Child route truy cập route data

```
export class DictionaryDetailComponent implements OnInit {  
  word: IWord;  
  constructor(  
    private activatedRoute: ActivatedRoute,  
    private dictionaryService: DictionaryService) { }  
  
  ngOnInit() {  
    const {snapshot} = this.activatedRoute;  
    const key = snapshot.paramMap.get('key');  
    const meaning = this.dictionaryService.search(key);  
    this.word = {key: key, meaning: meaning};  
  }  
}
```

Sử dụng service này để có thể lấy được param, query param, etc.

Từ đó có thể xử lý business cho component.

Snapshot cho phép chúng ta lấy ra thông tin cần thiết của route hiện tại

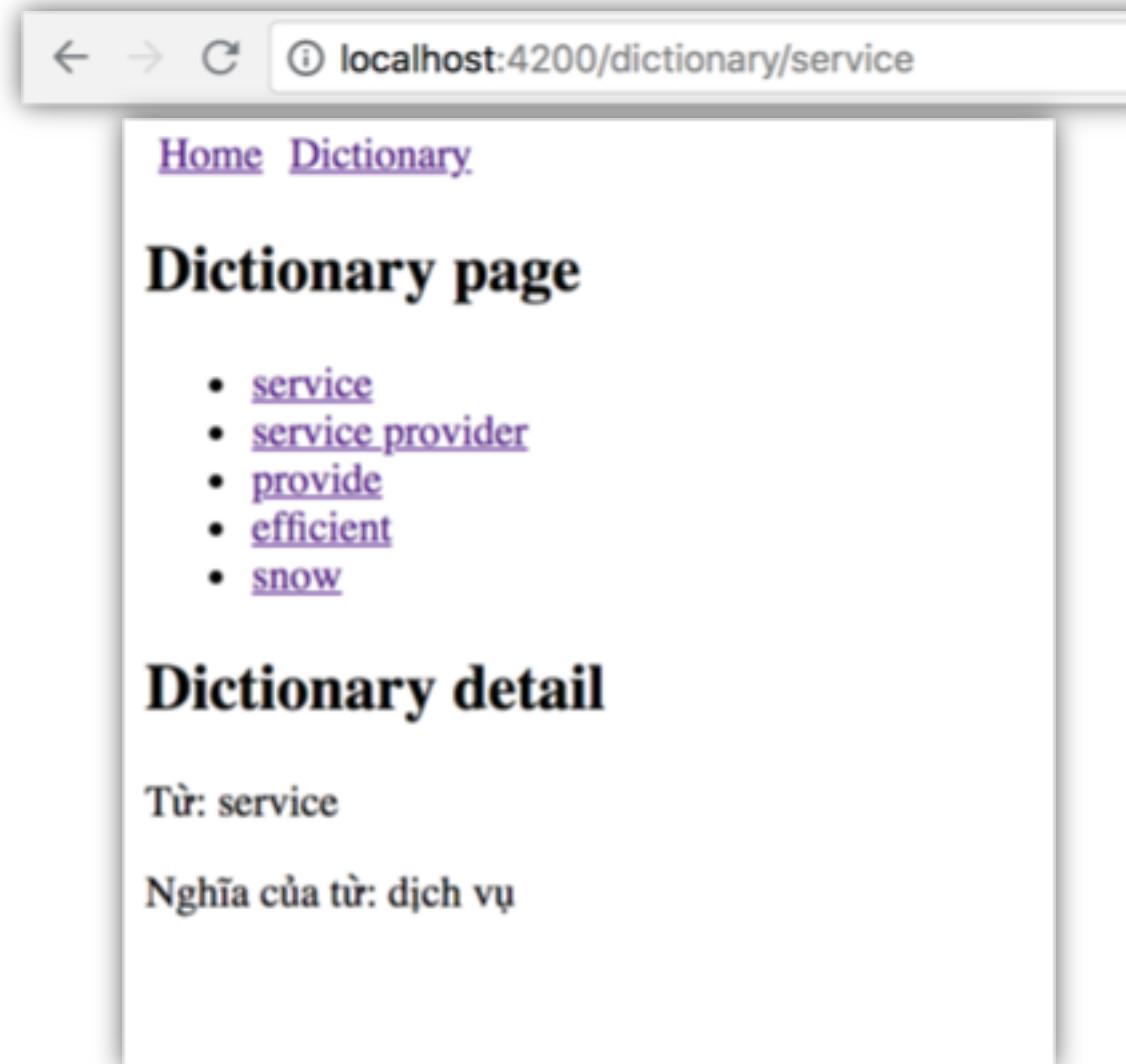
# Child route truy cập route data

<h2>Dictionary detail</h2>

<p>Từ: {{word.key}}</p>

<p>Nghĩa của từ: {{word.meaning}}</p>

# Child route truy cập route data



# ActivatedRoute

- Nếu bạn sử dụng snapshot, và bạn đang đứng ở page dictionary con, sau đó bạn di chuyển tiếp sang một page dictionary con khác, lúc này snapshot sẽ không update và view sẽ không được update.
- Giải pháp lúc này là dùng Observable có trong ActivatedRoute.

# Child route truy cập route data

```
export class DictionaryDetailComponent implements OnInit, OnDestroy {  
  sub: Subscription;  
  
  ngOnInit() {  
    this.sub = this.activatedRoute.paramMap.subscribe((paramMap: ParamMap) => {  
      const key = paramMap.get('key');  
      const meaning = this.dictionaryService.search(key);  
      this.word = {key: key, meaning: meaning};  
    });  
  }  
  ngOnDestroy(): void {  
    this.sub.unsubscribe();  
  }  
}
```

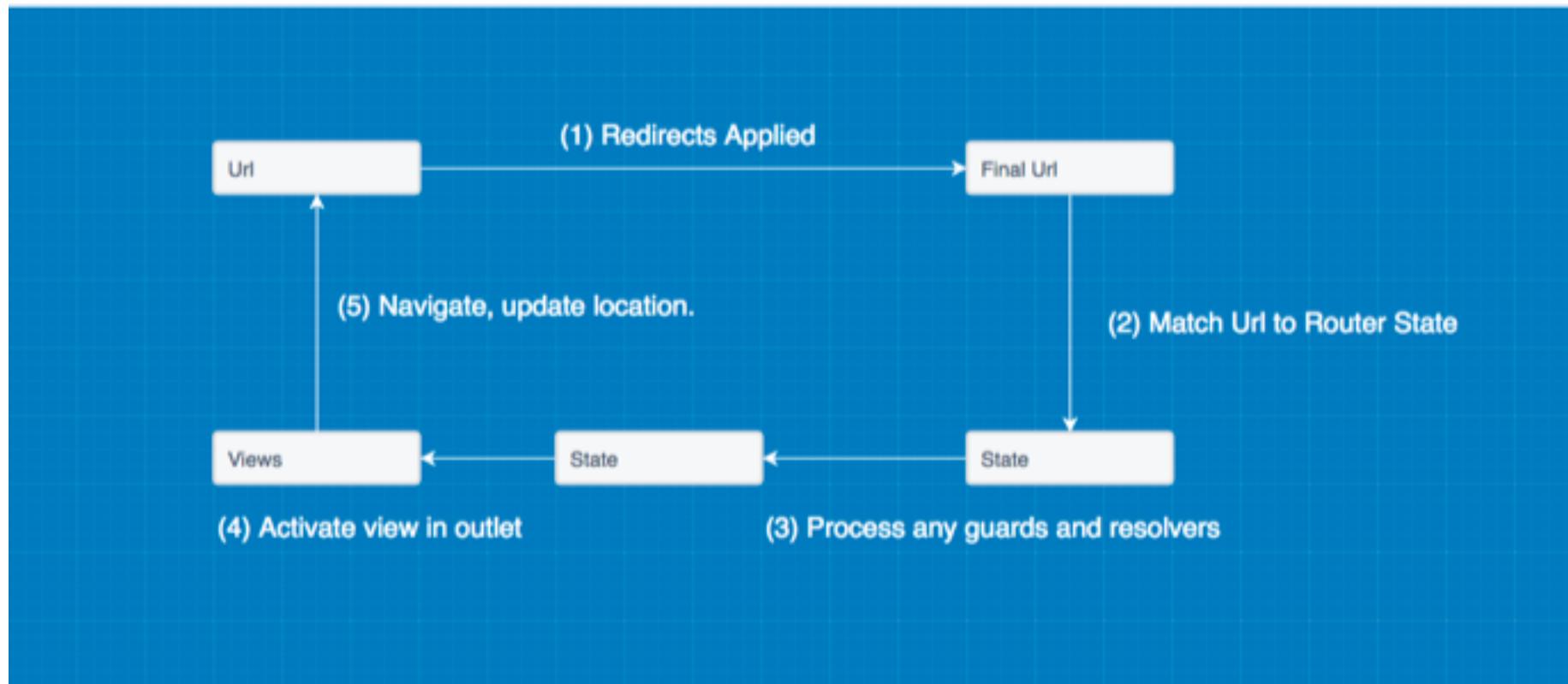
# Route guards

- Trong nhiều trường hợp, bạn muốn kiểm tra xem người dùng có được phép truy cập vào page đó không.
  - Giả sử page đó cần phải đăng nhập mới xem được.
  - Hoặc page đó cần có quyền admin mới vào được.
  - Hoặc app phải fetch data nào đó trước rồi mới vào page.
  - Hoặc app phải lưu thông tin rồi mới chuyển page.
  - Hoặc app phải hỏi người dùng xem có chắc chắn muốn chuyển page.
- Giải pháp lúc này là route guards.

# Route guards

- Angular router hỗ trợ bạn với nhiều loại guard khác nhau.
- CanActivate kiểm tra xem có được di chuyển tới target route không.
- CanActivateChild kiểm tra xem có di chuyển tới target child route không.
- CanDeactivate kiểm tra xem có được rời khỏi route hiện tại không.
- Resolve sử dụng để lấy dữ liệu trước khi vào route.
- CanLoad kiểm tra xem có được load lazy-load route không.

# Route lifecycle



# Route guards

- Tạo mới auth service, auth guard như sau:
  - o ng g s auth
  - o ng g guard auth

# Route guards

The image shows a code editor with two files open:

- auth.service.ts**:

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class AuthService {
7
8   constructor() {}
9 }
10
```
- auth.guard.ts**:

```
1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AuthGuard implements CanActivate {
9   canActivate(
10     next: ActivatedRouteSnapshot,
11     state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
12     return true;
13   }
14 }
```

# Route guards

The image shows a code editor with two tabs open. The left tab is titled 'auth.service.ts' and the right tab is titled 'auth.guard.ts'. Both tabs have a small icon of a person in a blue circle.

```
auth.service.ts ×
```

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class AuthService {
7   user: any;
8   constructor() {}
9   isAuthenticated(): boolean {
10   return !!this.user;
11 }
12 login() {
13   this.user = {
14     name: 'bob',
15     token: 'token'
16   };
17 }
18 logout() {
19   this.user = null;
20 }
21 }
```

```
auth.guard.ts ×
```

```
1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
3 import { Observable } from 'rxjs';
4 import { AuthService } from './auth.service';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class AuthGuard implements CanActivate {
10   constructor(private authService: AuthService) {}
11   canActivate(
12     next: ActivatedRouteSnapshot,
13     state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
14     return this.authService.isAuthenticated();
15   }
16 }
17
```

# Route guards

```
const routes: Routes = [
{
  path: 'dictionary',
  component: DictionaryPageComponent,
  children: [
    {
      path: ':key',
      component: DictionaryDetailComponent,
      canActivate: [AuthGuard]
    }
  ]
};
];
```

# Route guards

```
export class DictionaryPageComponent implements OnInit {  
    listWord: IWord[] = [];  
    constructor(  
        private dictionaryService: DictionaryService,  
        public authService: AuthService  
    ) {}  
  
    ngOnInit() {  
        this.listWord = this.dictionaryService.getAll();  
    }  
}
```

# Route guards – dictionary-page.component.html

```
<div>
  <button (click)="authService.login()">
    Login
  </button>
  <button (click)="authService.logout()">
    Logout
  </button>
</div>
```

# Route guards – dictionary-page.component.html

The screenshot shows a web page titled "Dictionary page". At the top left, there are navigation links: "Home" and "Dictionary". Below the title, there is a bulleted list of words: "service", "service provider", "provide", "efficient", and "snow". At the bottom left, there are two buttons: "Login" and "Logout". The page has a light gray background and a white content area.

Home Dictionary

## Dictionary page

- [service](#)
- [service provider](#)
- [provide](#)
- [efficient](#)
- [snow](#)

Login Logout

# Route guards

- Khi user chưa login, họ không thể di chuyển vào các page con.
- Lúc này guard đã phát huy tác dụng của nó.
- Sau khi user click vào nút login, họ lại có thể chuyển page bình thường.
- Mỗi khi bạn muốn sử dụng một guard nào đó, thì phải cài đặt method tương ứng.

# Truyền dữ liệu khi routing

- Sẽ làm thế nào để khi user di chuyển từ page này sang page khác mà page đích cần một số thông tin từ page trước?
- Giải pháp có thể sử dụng đến là service.
- Giả sử có màn hình đăng nhập gồm 2 bước, bước 1 nhập username, bước 2 nhập password và sử dụng 2 routes khác nhau.

# Truyền dữ liệu khi routing

- Tạo mới login service, login component như sau:
  - o ng g s login-service
  - o ng g c login-step1
  - o ng g c login-step2

# Login Service

```
@Injectable({  
  providedIn: 'root'  
})  
export class LoginServiceService {  
  loginData = {  
    username: '',  
    password: ''  
};  
  constructor() { }  
}
```

# Route config

```
{  
  path: 'login-step-1',  
  component: LoginStep1Component  
},  
{  
  path: 'login-step-2',  
  component: LoginStep2Component  
}
```

# Login Step 1 Component

```
<h2>Enter username</h2>

<div>
  <label>
    Username:
    <input type="text" [FormControl]="username">
  </label>
</div>
<button (click)="nextStep()">Next</button>
```

# Login Step 1 Component

```
class LoginStep1Component {  
  username = new FormControl('');  
  constructor(  
    private router: Router,  
    private loginSrv: LoginServiceService) { }  
  
  nextStep() {  
    this.loginSrv.loginData.username = this.username.value;  
    this.router.navigateByUrl('/login-step-2');  
  }  
}
```

Lưu trữ data vào service để dùng cho bước sau.

# Login Step 2 Component

```
<h2>Enter Password</h2>

<div>
  <label>
    Password for {{username}}:
    <input type="password" [formControl]="password">
  </label>
</div>
<button (click)="login()">Next</button>
```

# Login Step 2 Component

```
LoginStep2Component implements OnInit {  
  username: string;  
  password = new FormControl();  
  constructor(private loginSrv: LoginServiceService) {}  
  
  ngOnInit() {  
    this.username = this.loginSrv.loginData.username;  
  }  
  
  login() {  
    // do login  
  }  
}
```

Lấy data đã lưu trước đó trong service

# Kết quả thực hiện

## Enter username

Username: bob

Next

## Enter Password

Password for bob: ....

Next

# Lazy-load route

- Khi ứng dụng trở nên to hơn, chúng ta sẽ chia nhỏ thành các NgModule để dễ dàng tổ chức, bảo trì code.
- Router support chúng ta lazy-load từng NgModule theo yêu cầu.
- Giúp giảm việc tải bằng cách chia nhỏ thành các phần, và tải về từng phần.
- Rất hữu ích khi sử dụng trong các project thực tế.

# Lazy-load route

- Tạo mới NgModule và component như sau:
  - ng g m blog --routing
  - ng g c blog/blog-list

# Lazy-load route – blog-routing.module.ts

```
const routes: Routes = [  
  {  
    path: '',  
    component: BlogListComponent  
  }];  
  
@NgModule({  
  imports: [RouterModule.forChild(routes)],  
  exports: [RouterModule]  
})  
export class BlogRoutingModule { }
```

Do là feature module, nên chúng ta sử dụng static method forChild thay vì forRoot.

# Lazy-load route – app-routing.module.ts

```
const routes: Routes = [ {  
    path: 'blog',  
    loadChildren: './blog/blog.module#BlogModule'  
}];
```

Sử dụng loadChildren, và trả tới đường dẫn tương đối của file so với file hiện tại

# Lazy-load route – app.component.html

```
<nav>
  <a routerLink="/" class="link">Home</a>
  <a routerLink="dictionary" class="link">
    Dictionary
  </a>
  <a routerLink="blog" class="link">Blog</a>
</nav>
```

# Lazy-load route

- BlogListComponent hiện tại bạn có thể tìm thấy trong folder src/app/blog/blog-list, nó chỉ là một dump component để minh họa.
- Khi người dùng click vào link để chuyển trang sang trang blog, app lúc này mới bắt đầu tải phần code tương ứng về.

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2018-09-09T04:07:45.725Z
Hash: b809d735ffdb765f440f
Time: 8284ms
chunk {blog-blog-module} blog-blog-module.js, blog-blog-module.js.map (blog-blog-module) 22.7 kB [rendered]
chunk {main} main.js, main.js.map (main) 40.7 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 8 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.76 MB [initial] [rendered]
i [wdm]: Compiled successfully.
```

# Welcome to angular-service-router!



## English-Vietnamese Dictionary

---

Home Dictionary Blog

home works!

Network

Filter Hide data URLs All XHR JS C

Name	Status	Type
runtime.js	200 OK	script
polyfills.js	200 OK	script
styles.js	200 OK	script
vendor.js	200 OK	script
main.js	200 OK	script
ng-validate.js	200 OK	script
backend.js	200 OK	script

# Welcome to angular-service-router!



## English-Vietnamese Dictionary

[Home](#) [Dictionary](#) [Blog](#)

### Blog List

Quis, nesciunt delectus cupiditate quam incident illum ipsa repellat voluptatibus ab suscipit nostrum! Odio explicabo consequuntur iste debitis hic iusto, voluptatibus, repellat autem suscipit nihil minima magni voluptate corporis quasi illum aspernatur tempora eaque, ipsa quo fugit ea? Alias cupiditate culpa doloremque modi, ipsum dolores eos facilis provident quos eligendi nisi dicta ad? Similique cumque impedit aliquid fugiat dolorum ex vel fugit distinctio, repellendus, sed edit. Autem quis quas aliquid molestias ipsa id deleniti fugiat maxime mollitia nostrum amet, vero aperiam non aliquam velit neque, labore delectus animi harum incident similique beatiae ratione voluptatum excepturi! Iure voluptatum blanditiis in harum nulla, ea deserunt provident voluptatibus, sequi unde omnis aut magni nobis natus architecto dolores doloribus eos aliquam perspicatis veniam. Incident, adipisci ex. Facere ad, eveniet cumque et, commodi pariatur quod amet vel fuga porro aspernatur, soluta perspicatis. Dignissimos, molestias ex fugiat voluptates atque obcaecati. Eorum facere cupiditate, reprehenderit deleniti repellendus ex laborum querat dolorum reiciendis aliquam a voluptatem repudiandae preferendis impedit illum animi nostrum numquam illo optio soluta facilis accusamus nemo id. Eorum sunt, distinctio omnis, facere voluptatem debitis dignissimos pariatur et ipsa libero, nulla asperiores corrupti? Autem deleniti ex exercitationem et omnis dolor voluptatum ratione, tenetur non nesciunt dignissimos, labore sint eos distinctio? Temporibus saepe magni tempore obcaecati quidem explicabo necessitatibus cupiditate officiis perspicatis deserunt iusto ratione, rem dignissimos repellendus querat sequi dolores vel iste, cum excepturi, eum incident ipsum nostrum furevit! Elizendi esse, cum resident, nesciunt facilis, iste, reiciendis, sed.

Elements Console Sources Network **Perf**

Filter Hide data URLs All XHR JS

Name	Status	Type
runtime.js	200 OK	script
polyfills.js	200 OK	script
styles.js	200 OK	script
vendor.js	200 OK	script
main.js	200 OK	script
ng-validate.js	200 OK	script
backend.js	200 OK	script
<b>blog-blog-module.js</b>	200 OK	script

# Lazy-load route

- Vấn đề về mạng và size của phần đó, có thể khiến người dùng phải chờ đợi. Lúc này Angular cho phép chúng ta thiết lập cơ chế Preload để tải về các phần có thể tải, sau khi app đã khởi tạo.
- Để cài đặt Preload, bạn có thể sử dụng ngay công cụ mà Angular cung cấp như sau.

# Lazy-load route

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

localhost:4200

# Welcome to angular-service-router!



## English-Vietnamese Dictionary

Home Dictionary Blog

home works!

Network

View: Group by frame Preserve log Disable cache Offline Online

Name	Status	Type	Initiator	Size	Time	Waterfall
runtime.js	200 OK	script	(index)	8.1 KB	15 ms	
polyfills.js	200 OK	script	(index)	222 KB	18 ms	
styles.js	200 OK	script	(index)	15.9 KB	17 ms	
vendor.js	200 OK	script	(index)	3.6 MB	206 ms	
main.js	200 OK	script	(index)	40.2 KB	27 ms	
blog-blog-m...	200 OK	script	bootstrap:...	22.4 KB	5 ms	
			Script	22.2 KB	5 ms	

# Link tham khảo

- <https://angular.io/guide/architecture-services>
- <https://angular.io/guide/router>
- <https://blog.angularindepth.com/the-three-pillars-of-angular-routing-angular-router-series-introduction-fb34e4e8758e>

---

CODEGYM

CODEGYM

R a i s i n g t h e b a r