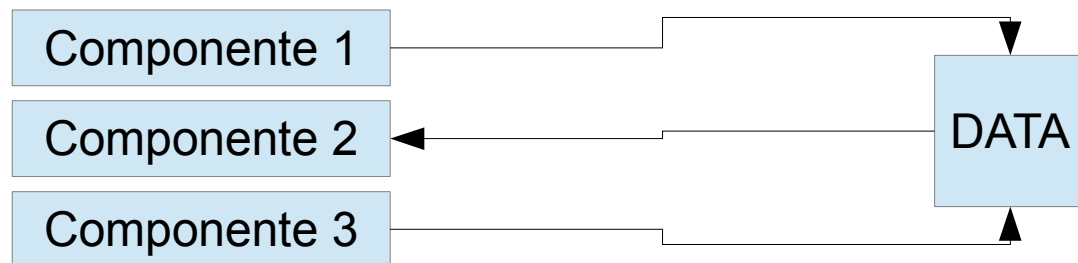


Projeto de Interfaces WEB

Serviços do Angular Aula 04

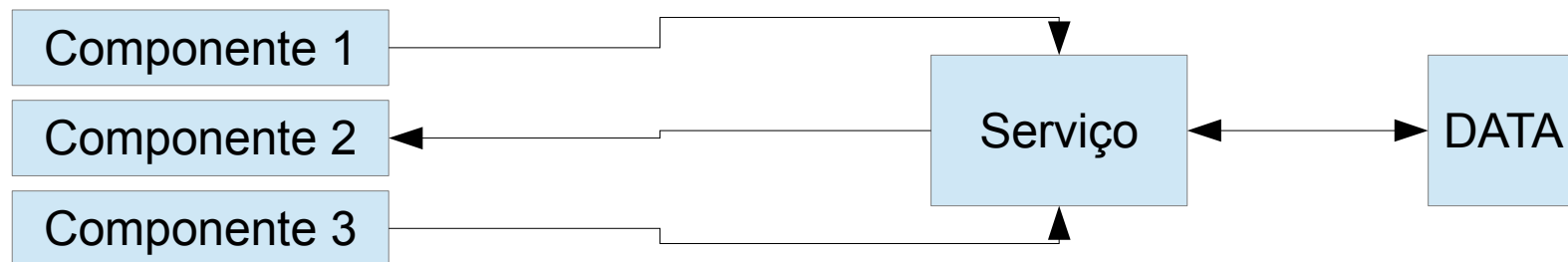
Introdução

- O que são serviços?
 - São funcionalidades que podem ser compartilhadas entre vários comportamentos.
 - Por exemplo, o acesso aos dados em um sistema.



Introdução

- Solução:
 - O ideal é criar um componente “intermediário” que irá fazer o acesso aos dados.



Implementando um Serviço

- Volte a este estado no seu projeto. Não esqueça do bootstrap.

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

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {  
  nome = "Jefferson de Carvalho";
```

```
  salvar(nomeInput:string){  
    this.nome = nomeInput;  
  }  
}
```

```
<div class="container">
```

```
  <div class="alert alert-primary" role="alert">  
    Seu nome é {{nome}}  
  </div>
```

```
  <div class="form-group">  
    <label>Nome</label>  
    <input type="text" class="form-control"  
#nomeInput>  
  </div>
```

```
  <button type="button" class="btn btn-primary"  
(click)="salvar(nomeInput.value)">Salvar</button>  
</div>
```

Implementando um Serviço

- Crie o arquivo pessoa.service.ts, dentro de app.

```
export class PessoaService{

  contadorId:number = 1;
  pessoas:any[] = [{id:1,nome:'Jefferson'}];

  adicionar(nome:string){
    this.contadorId++;
    const pessoa = {id:this.contadorId,nome:nome};
    this.pessoas.push(pessoa);
    console.log(this.pessoas);
  }

  consultar(){
    return this.pessoas;
  }
}
```

Implementando um Serviço

- Como usar a classe de serviço (forma ingênua)?

```
export class AppComponent {  
  nome = "Jefferson de Carvalho";  
  pessoaService:PessoaService;  
  
  constructor(){  
    this.pessoaService = new PessoaService();  
  }  
  
  salvar(nomeInput:string){  
    this.pessoaService.adicionar(nomeInput);  
    this.nome = nomeInput;  
  }  
}
```

Referência ao serviço.

Criando o objeto.

Usando.

Implementando um Serviço

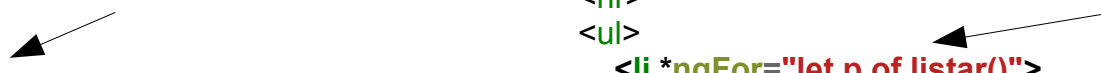
- Exercício:
 - Agora, usando o `*ngFor`, liste os funcionários a partir do objeto de serviço.

Implementando um Serviço

- Solução

.ts

```
export class AppComponent {  
  nome = "";  
  pessoaService:PessoaService;  
  
  constructor(){  
    this.pessoaService = new PessoaService();  
  }  
  
  salvar(nomeInput:string){  
    this.pessoaService.adicionar(nomeInput);  
    this.nome = nomeInput;  
  }  
  
  listar(){  
    return this.pessoaService.consultar();  
  }  
}
```



.html

```
<div class="container">  
  
  <div class="alert alert-primary" role="alert">  
    Seu nome é {{nome}}  
  </div>  
  
  <div class="form-group">  
    <label>Nome</label>  
    <input type="text" class="form-control" #nomeInput>  
  </div>  
  
  <button type="button" class="btn btn-primary"  
    (click)="salvar(nomeInput.value)">Salvar</button>  
  
  <hr>  
  <ul>  
    <li *ngFor="let p of listar()">  
      {{p.nome}}  
    </li>  
  </ul>  
</div>
```


Implementando um Serviço

- E se criarmos dois componentes para tarefas específicas?
Por exemplo:
 - Crie o componente pessoa-form
 - **ng g c pessoa-form --spec=false**
 - Crie o componente pessoa-list
 - **ng g c pessoa-list --spec=false.**

Implementando um Serviço

- Exercício:
 - Coloque o código html referente a inserção de uma pessoa em `pessoa-form.html`
 - Coloque o código referente a listagem de pessoas dentro de `pessoa-list.html`.
 - Faça as as correções dentro dos respectivos `.ts`

Implementando um Serviço

- Em pessoa-form:

.ts

```
import { PessoaService } from '../pessoa.service';
import { Component } from '@angular/core';

@Component({
  selector: 'app-pessoa-form',
  templateUrl: './pessoa-form.component.html',
  styleUrls: ['./pessoa-form.component.css']
})
export class PessoaFormComponent {

  nome = '';
  pessoaService: PessoaService;

  constructor(){
    this.pessoaService = new PessoaService();
  }

  salvar(nomeInput: string){
    this.pessoaService.adicionar(nomeInput);
    this.nome = nomeInput;
  }
}
```

.html

```
<div class="alert alert-primary" role="alert">
  Seu nome é {{nome}}
</div>

<div class="form-group">
  <label>Nome</label>
  <input type="text" class="form-control" #nomeInput>
</div>

<button type="button" class="btn btn-primary"
  (click)="salvar(nomeInput.value)">Salvar</button>
```

Implementando um Serviço

- Em pessoa-list

.ts

```
import { PessoaService } from './../pessoa.service';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-pessoa-list',
  templateUrl: './pessoa-list.component.html',
  styleUrls: ['./pessoa-list.component.css']
})
export class PessoaListComponent {

  nome = '';
  pessoaService: PessoaService;

  constructor(){
    this.pessoaService = new PessoaService();
  }

  listar(){
    return this.pessoaService.consultar();
  }
}
```

.html

```
<hr>
<ul>
  <li *ngFor="let p of listar()">
    {{p.nome}}
  </li>
</ul>
```

Implementando um Serviço

- Em `app.component.html`, basta chamar os dois seletores.
- Será que irá funcionar? Será que se eu adicionar uma pessoa via input, irá afetar a lista de pessoas, exibida pelo `*ngFor` de `pessoa-list`?
- Por quê?

Injeção de Dependência

- Gerenciar dependências e componentes de softwares.
- O Angular irá então “gerenciar” a criação de um objeto do tipo **PessoaService**.
- Faça as seguintes alterações:

```
constructor(private pessoaService:PessoaService){ }
```

Injeção de Dependência

- Em app.module.ts:

```
@NgModule({  
  declarations: [  
    AppComponent,  
    ColoridoDirective,  
    PessoaFormComponent,  
    PessoaListComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
  providers: [PessoaService],  
  bootstrap: [AppComponent]  
})
```



Angular cria o objeto pra gente.

Injeção de Dependência

- Forma literal:

```
@NgModule({  
  declarations: [  
    AppComponent,  
    ColoridoDirective,  
    PessoaFormComponent,  
    PessoaListComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
  providers: [  
    {provide: PessoaService, useClass: PessoaService}  
  ],  
  bootstrap: [AppComponent]  
})
```

Chave ou token.

Classe que será instanciada.

Injeção de Dependência

- Mas, por que a forma literal?
 - Quando temos mais de um classe filha, pode decidir qual objeto o Angular poderá instanciar pra gente.
 - Crie um classe filha de PessoaService (no mesmo arquivo):


```
export class PessoaGrandeService extends PessoaService{  
    adicionar(nome:string){  
        super.adicionar(nome.toUpperCase());  
    }  
}
```

Injeção de Dependência

- Agora, em app.module.ts:

```
@NgModule({  
  declarations: [  
    AppComponent,  
    ColoridoDirective,  
    PessoaFormComponent,  
    PessoaListComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
  providers: [  
    {provide: PessoaService, useClass: PessoaGrandeService}  
  ],  
  bootstrap: [AppComponent]  
})
```

Agora posso escolher uma implementação mais específica.



Trabalhando com Fábricas

- Em app.module.ts

```
const criarPessoaService = () =>{  
  return new PessoaGrandeService();  
}
```

Método de fábrica



```
@NgModule({  
  declarations: [  
    AppComponent,  
    ColoridoDirective,  
    PessoaFormComponent,  
    PessoaListComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
  providers: [  
    {provide: PessoaService, useFactory: criarPessoaService}  
  ],  
  bootstrap: [AppComponent]  
})
```

Usando o método de fábrica



Trabalhando com Fábricas

- Exercício:
 - Passe um parâmetro “tratamento:string” para o construtor de PessoaGrandeService e o concatene com o nome.
 - Por exemplo, o tratamento poderia ser “Sr. ”

Trabalhando com Fábricas

- Em app.module.ts e em PessoaGrandeService

```
const criarPessoaService = () =>{  
  return new PessoaGrandeService("Mestre");  
}
```

```
export class PessoaGrandeService  
  extends PessoaService{  
  
  constructor(private tratamento:string){  
    super();  
  }  
  
  adicionar(nome:string){  
    super.adicionar(this.tratamento + " "  
+ nome.toUpperCase());  
  }  
}
```

Usando o @Inject

- Apague o método de fábrica criado anteriormente e em app.module.ts, faça:

```
@NgModule({  
  declarations: [  
    AppComponent,  
    ColoridoDirective,  
    PessoaFormComponent,  
    PessoaListComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
  providers: [  
    {provide: PessoaService, useClass: PessoaGrandeService},  
    {provide: 'tratamento', useValue: "M.Sc."}  
  ],  
  bootstrap: [AppComponent]  
})
```

Volte a usar o “useClass”



Crie um objeto que “ligue” a string ‘tratamento’ com o valor ‘M.Sc’.



Usando o @Inject

- Em PessoaGrandeService

```
export class PessoaGrandeService extends PessoaService{

    constructor(@Inject('tratamento') private tratamento:string){
        super();
    }

    adicionar(nome:string){
        super.adicionar(this.tratamento + " " + nome.toUpperCase());
    }
}
```

Um serviço dentro de outro

- Crie um novo serviço com o comando:
 - **ng g s log --spec=false**
 - **Não é preciso mais colocar o serviço no Providers!**

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

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

```
export class LogService {
```

```
  log(msg:string){  
    console.log(`LOG: ${msg}`);  
  }  
}
```

→ O Angular já injeta automaticamente pra gente através do @Injectable.

Um serviço dentro de outro

- Em PessoaGrandeService:

Todo serviço que recebe uma “injeção” deve ter o `@Injectable`. Nas novas versões do Angular isso já é feito de forma automática.

```
@Injectable()
export class PessoaGrandeService extends PessoaService{

  constructor(@Inject('tratamento') private tratamento:string,
              private log:LogService){
    super();
  }

  adicionar(nome:string){
    this.log.log(`Adicionando pessoa ${nome}`);
    super.adicionar(this.tratamento + " " + nome.toUpperCase());
  }
}
```

Objeto a ser injetado. Deve ser declarado em **app.module.ts** (ou usando o `@Injectable`)

```
providers: [
  {provide: PessoaService, useClass:
    PessoaGrandeService},
  {provide:'tratamento',useValue: 'M.Sc.'},
  LogService
],
```