Aula 4

Desenvolvimento Web – Front End

Prof. Mauricio Antonio Ferste

1

Conversa Inicial

2

Resumo

- Este estudo trata de vários assuntos necessários para o entendimento geral do nosso curso
- Vamos repassar rapidamente esses tópicos, pois são estruturas muito importantes para o entendimento de padrões web

3

- Tema 1 - Injeção de dependências
- Tema 2 - Comunicação publisher-subscriber
- Tema 3 - Observables e subscribe
- Tema 4 - Formulários e reactive forms
- Tema 5 - Gerenciamento e tratamento de erros — handlers

4

Injeção de Dependências

5

Injeção de dependências

- Injeção de dependência é uma técnica para livrar ou remover o acoplamento entre os objetos e seus colaboradores ou dependentes
- Em vez de instanciar objetos colaboradores diretamente, ou usar referências estáticas, esses colaboradores são fornecidos para a classe dependente de um modo particular

6

Injeção de dependências

```
import { InternetModel} from '../model/internetModel'
import { Injectable } from '@angular/core';
import { BaseService } from '../base/base.service';
import { HttpService } from '../http.service';
import { IResultHttp } from '../interfaces/IResultHttp';
import { environment } from '../../../environments/environment';
import { Observable, Subject } from 'rxjs';

@Injectable({
 providedIn: 'root'
})
export class InternetService extends BaseService<InternetModel> {
 private loginSubject = new Subject<boolean>();
 constructor(public override http: HttpService) {
  super('internet', http);
 }
}
```

Fonte: Leandro Bianch, 2023

7

Injeção de dependências

- Estamos aplicando fundamentos básicos de Orientação a Objetos, como herança
- ....extends....

```
import { InternetModel} from '../model/internetModel'
import { Injectable } from '@angular/core';
import { BaseService } from '../base/base.service';
import { HttpService } from '../http.service';
import { IResultHttp } from '../interfaces/IResultHttp';
import { environment } from '../../../environments/environment';
import { Observable, Subject } from 'rxjs';

@Injectable({
 providedIn: 'root'
})
export class InternetService extends BaseService<InternetModel> {
 private loginSubject = new Subject<boolean>();
 constructor(public override http: HttpService) {
  super('internet', http);
 }
}
```

Fonte: Leandro Bianch, 2023

8

- @NgModule
- Em app.module.ts

Fonte: Leandro Bianch, 2023

9

Comunicação Publisher-Subscriber

10

EventEmitter

```
const EventEmitter = require('events');

// Criar uma nova instância do EventEmitter
const meuEmitter = new EventEmitter();

// Registrar um ouvinte de evento
meuEmitter.on('meuEvento', (arg) => {
 console.log('Evento ocorreu com argumento:', arg);
});

// Emitir o evento
meuEmitter.emit('meuEvento', 'Olá, mundo!');
```

11

EventEmitter no Angular

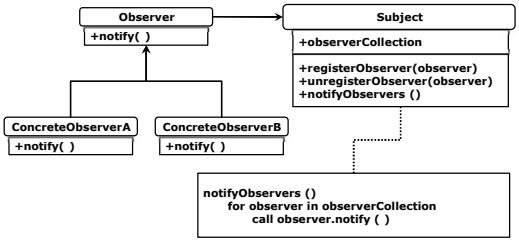
```
import { EventEmitter } from '@angular/core';

export class MeuComponente {
 meuEvento = new EventEmitter<string>();
}

his.meuEvento.emit('Olá, mundo!');

export class OutroComponente {
 constructor() {
  meuComponente.meuEvento.subscribe((valor: string) => {
   console.log('Evento ocorreu com argumento:', valor);
  });
 }
}
```

12

Observables e Angular

13

Padrão Observer



14

Um exemplo prático

```
import { Observable } from 'rxjs';

const meuObservable = new Observable((observer) => {
  observer.next('Primeiro valor');
  observer.next('Segundo valor');
  observer.next('Terceiro valor');
  observer.complete();
});
```

15

Um exemplo prático

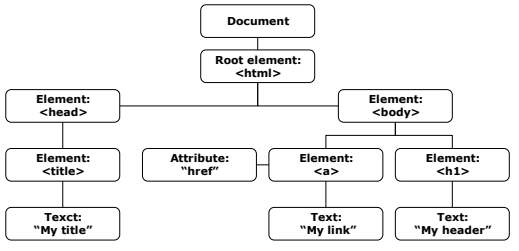
```
meuObservable.subscribe(
  (valor) => {
    console.log('Valor emitido:', valor);
  },
  (erro) => {
    console.error('Erro:', erro);
  },
  () => {
    console.log('Observable completo.');
  }
);
```

16

Formulários e Reactive Forms

17

Árvore

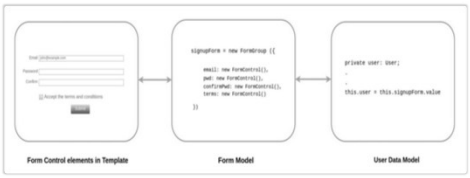


18

Reactive Forms

19

Reactive Forms



20

Gerenciamento e Tratamento de Erros - Handlers

21

Tratamento de Retornos

```
private handleError(error: HttpErrorResponse) {
if (error.status === 0) {
  // A client-side or network error occurred. Handle it accordingly.
  console.error('An error occurred:', error.error);
} else {
  // The backend returned an unsuccessful response code.
  // The response body may contain clues as to what went wrong.
  console.error(
    `Backend returned code ${error.status}, body was: `, error.error);
}
  // Return an observable with a user-facing error message.
return throwError(() => new Error('Something bad happened; please try again
later.'));
}
```

22

Tratamento de Retornos

```
public get(url: string): Promise<IResultHttp> {
  const header = this.createHeader();
  console.log("header " + header)
  console.log(url)
  return new Promise(async (resolve) => {
   try {
    const res = await this.http.get(url, { headers: header }).toPromise();
    console.log("aqui")
    resolve({ success: true, data: res, error: undefined });
  } catch (error) {
    resolve({ success: false, data: {}, error });
   }
  });
}
```

23