

Entendendo a Estrutura do Angular

Agenda

- Introdução ao Angular
- Componentes no Angular
- Módulos Angular
- Serviços e Injeção de Dependência
- Decorações no Angular
- Diretivas
- Template e Dados
- Roteamento e Navegação
- Teste de conhecimento
- Conclusão e Próximos Passos

Introdução ao Angular

O Angular é um framework de código aberto desenvolvido pelo Google para a criação de aplicativos da web.

Ele utiliza a linguagem TypeScript, que é uma variação do JavaScript, para escrever o código.

O Angular oferece uma arquitetura modular e componentes reutilizáveis para facilitar o desenvolvimento de aplicativos escaláveis e de alto desempenho.

Componentes no Angular

O que são componentes?

- Componentes são a unidade básica de construção de uma aplicação Angular.
- Eles são responsáveis por controlar a lógica de negócio e renderizar a interface do usuário.
- Cada componente é composto por um template HTML, uma classe TypeScript e um arquivo de estilo.

Benefícios dos componentes

- Componentes são a unidade básica de construção de uma aplicação Angular.
- Eles são responsáveis por controlar a lógica de negócio e renderizar a interface do usuário.
- Cada componente é composto por um template HTML, uma classe TypeScript e um arquivo de estilo.

Componentes (Components)

typescript

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  // Lógica do componente
}
```

Módulos Angular

O que são módulos?

Os módulos são um mecanismo de organização e encapsulamento de funcionalidades em uma aplicação Angular.

Eles agrupam componentes, diretivas, serviços e outros recursos relacionados, facilitando a reutilização e a manutenção do código.

Permitem a criação de escopos de injeção de dependências.

Benefícios dos módulos

Promovem a modularidade e a separação de preocupações, permitindo que a aplicação seja dividida em partes menores e mais gerenciáveis.

Facilitam a colaboração entre desenvolvedores em grandes projetos, pois cada módulo pode ser desenvolvido independentemente.

Permitem a lazy loading, ou seja, o carregamento sob demanda de módulos específicos, melhorando o desempenho da aplicação.



Plus tip:

Proporcionam um ambiente isolado para cada módulo, evitando conflitos de nomes e facilitando a resolução de dependências.

Módulos (Modules)

typescript

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Serviços e Injeção de Dependência

1

Um serviço é uma classe que pode ser injetada em outros componentes ou serviços para fornecer funcionalidades reutilizáveis.

2

O Angular possui um mecanismo de injeção de dependência embutido que gerencia a criação e a injeção de serviços.

3

A injeção de dependência é um padrão de design que permite que as dependências sejam injetadas automaticamente em um componente ou serviço.

Serviços (Services)

typescript

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

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

```
export class MyService {  
  // Lógica do serviço  
}
```

Decorator no Angular

O que são decorator ?



As decorações são recursos do Angular que permitem adicionar metadados a classes, métodos e propriedades. Elas são usadas para alterar o comportamento do Angular e adicionar recursos extras.

Exemplos de decorato



Algumas das decorações mais comuns no Angular incluem: @Component, @Directive, @Input, @Output, @ViewChild, @HostListener.

Benefícios dos decorato



As decorações no Angular oferecem os seguintes benefícios: melhor organização do código, facilidade de leitura e compreensão, maior reutilização de código e capacidade de adicionar comportamentos extras.

Decorator

```
function log(target: any) {  
  console.log(target);  
}
```

```
@log  
class Foo {}
```

```
[LOG]: class Foo { }
```

```
function log(prefix: string) {  
  return (target: any) => {  
    console.log(`${prefix} - ${target}`);  
  }  
}
```

```
@log('Awesome')  
class Foo {}
```

```
[LOG]: "Awesome - class Foo { }"
```

Diretivas

A vertical dotted line runs down the left side of the slide. It features three colored dots: a teal one at the top, a blue one in the middle, and a purple one at the bottom. Each dot has a short horizontal bar extending to the right.

O que são diretivas?

Diretivas são um recurso do Angular que permitem estender o HTML com comportamentos personalizados.

Tipos de diretivas

Existem três tipos de diretivas no Angular: diretivas de atributo, de elemento e estruturais.

Exemplos de uso

Diretivas populares no Angular incluem `ngIf`, `ngFor`, `ngClass` e `ngStyle`.

Diretivas (Directives)

```
<div *ngFor="let item of items">{{ item }}</div>
```

```
<div *ngIf="true">{{ item }}</div>
```

```
<div *[hidden]="true">{{ item }}</div>
```

```
<div [ngClass]="step == 'step1' ? 'my_class1' : 'my_class2'"></div>
```

Template e Dados




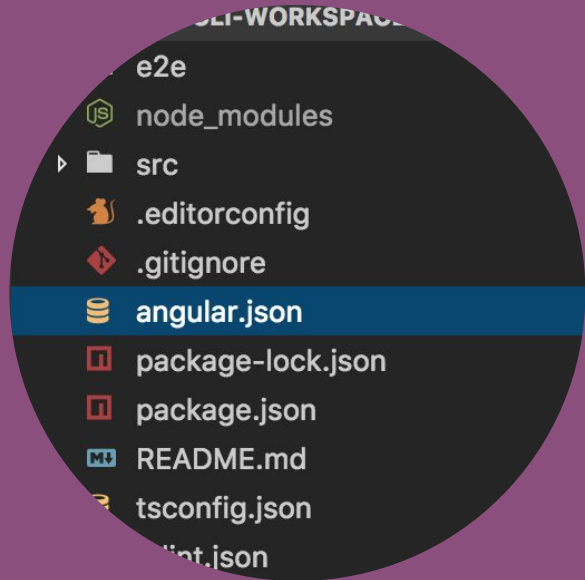
Templates

- Os templates são responsáveis por definir a estrutura e a aparência da interface do usuário.
- Eles combinam elementos HTML com diretivas e expressões do Angular para renderizar dinamicamente os dados.
- Os templates podem conter diretivas estruturais, como `ngFor` e `ngIf`, para iterar sobre listas e exibir conteúdo condicionalmente.
- Além disso, os templates podem ser reutilizados em diferentes componentes, tornando o desenvolvimento mais eficiente.

Roteamento e Navegação

O que é o roteamento?

- O roteamento é um recurso do Angular que permite navegar entre diferentes componentes e exibir diferentes conteúdos em uma única página da web.
 - Ele é utilizado para criar aplicativos de várias páginas, onde cada página é um componente separado.
 - O roteamento é gerenciado por um módulo chamado RouterModule, que define as rotas disponíveis e suas respectivas URLs.
-  **Plus tip:**
- As rotas podem ser configuradas para exibir diferentes componentes e passar parâmetros para eles.



Roteamento

```
const routes: Routes = [  
  { path: 'home', component: HomeComponent },  
  { path: 'about', component: AboutComponent },  
  { path: '**', redirectTo: '/home' }  
];
```


Conclusão e Próximos Passos

1

Nesta apresentação, você aprendeu os conceitos fundamentais do Angular e como criar aplicações web poderosas.

2

Recomendamos que você continue a aprender e aprofundar seus conhecimentos em Angular através de cursos online, documentação oficial e projetos práticos.

3

Agora que você está familiarizado com o Angular, é hora de praticar e explorar mais recursos.