UNIPAR

Semana Acadêmica 2022

Aula 01

Desenvolvendo um Todo List com Angular

Henrique Augusto Maltauro

Henrique Augusto Maltauro

Formação Acadêmica

- ***Graduação (2016 - 2018) - Unipar:*** Tecnologia em Análise e Desenvolvimento de Sistemas
- ***Pós-Graduação (2021 - Atualmente) - Unipar:*** Especialização em Desenvolvimento de Aplicações Para WEB e Mobile

Experiência Profissional

- ***Inside Sistemas (2016 - 2016):*** Estágio Suporte
- ***InterSystem Sistemas (2016 - 2017):*** Desenvolvedor Desktop
- ***Junsoft Sistemas (2017 - 2020):*** Desenvolvedor Desktop
- ***Maxicon Sistemas (2020 - 2020):*** Desenvolvedor Desktop
- ***Inside Sistemas (2021 - 2021):*** Desenvolvedor Mobile
- ***Metadados Sistemas de RH (2022 - Atualmente):*** Desenvolvedor Full Stack

Angular

Angular

Angular é um framework para desenvolver aplicações web, mobile e desktop, mantido e desenvolvido pela Google

Ele vem com um conjunto de bibliotecas poderosas que podemos importar, possibilitando construir aplicações com uma qualidade e produtividade surpreendente

Angular

A primeira versão foi lançada em 2009, sobre o nome de AngularJs

Seguia a filosofia de programação declarativa, estendendo o HTML para facilitar a criação de interfaces com conteúdo dinâmico

Angular

Com o passar dos anos o HTML, CSS e JavaScript foram evoluindo, e se percebeu que o AngularJs não conseguia acompanhar essa evolução e manter a performance ao mesmo tempo

Assim foi lançado em 2016 uma versão nova do Angular, a qual foi reescrita totalmente do zero e focada em TypeScript

Angular

Inicialmente a reescrita não foi muito aceita pela comunidade, principalmente por dois fatores

- Uma reescrita tornava as aplicações feitas na versão antiga, incompatíveis com a nova
- Na época o TypeScript não era muito aceito

Angular

AngularJS teve o suporte finalizado em Janeiro de 2022, e o framework ficou conhecido apenas como Angular

Uma nova versão do Angular é lançada a cada mais ou menos 6 meses, mas aqui é um atualização, e não uma reescrita total

Angular CLI

Angular CLI

Command-Line Interface

É uma ferramenta usada para inicializar, desenvolver, escalonar e manter aplicações Angular diretamente de um shell de linha de comando

O Angular CLI é um pacote NodeJs

Angular CLI

A instalação do Angular CLI é realizada através do comando `npm install -g @angular/cli`

Todos os comandos do Angular CLI são invocados através do comando `ng`

Angular CLI

```
ng version
ng v
```

- Verifica a versão do Angular CLI instalada na máquina

Angular CLI

```
ng new [project-name]
ng n [project-name]
```

- Cria um workspace angular totalmente do zero, com um projeto dentro e inicializa o projeto, adicionando todos os arquivos necessários para executar o projeto

Angular CLI

```
ng serve [project-name]
ng s [project-name]
```

- Executa um build da aplicação e inicia um servidor http local na porta 4200, para que a aplicação possa ser executada
- Toda vez que tiver uma alteração, vai ser executado novamente um build da aplicação sem reiniciar o servidor

Angular CLI

```
ng generate <schematic>
ng g <schematic>
```

- Vai gerar e atualizar arquivos, baseados em um schema, prontos para funcionarem de acordo com o schema utilizado
- Ex: Componente, Biblioteca, Classe, Interface, Módulo, Projeto, etc

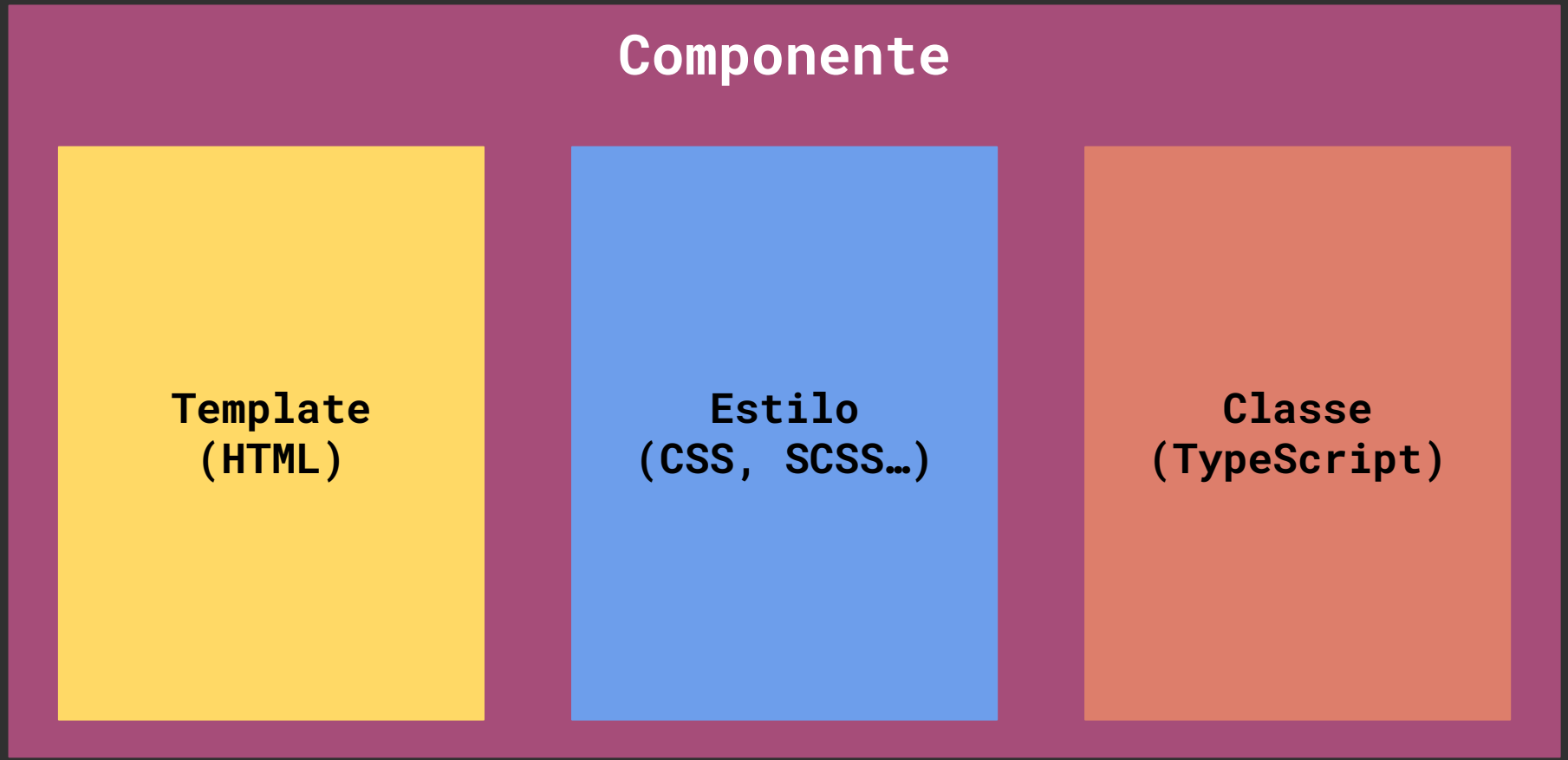
Componentes

Componentes

Uma aplicação Angular é baseada em componentes, possibilitando o agrupamento de comportamento e estilização

Componentes permitem dividir a interface do usuário em partes independentes e reutilizáveis

Os componentes podem receber valores, podem retornar valores e podem ter outros componentes dentro deles


Componente
Template
(HTML)
Estilo
(CSS, SCSS…)
Classe
(TypeScript)

Componentes

```
ng generate component [component-name]
ng g c [component-name]
```

- Vai gerar e atualizar os arquivos necessários para criar um componente genérico totalmente pronto para funcionar

Ciclo de Vida dos Componentes

Ciclo de Vida dos Componentes

Os componentes possuem um ciclo de vida, ou seja, eles possuem uma série de eventos pré-definidos que são executados em uma ordem específica

Ao total são 8 estágios no ciclo de vida de um componentes

Podemos usar esses eventos, nos diferentes estágios do componente, para obter controle sobre o componente

Ciclo de Vida dos Componentes

`constructor()`

- Como o Angular é estruturado em Orientação a Objeto, a classe do componente possui um construtor, que vai ser executado na construção do componente e antes de qualquer estágio do ciclo de vida

Ciclo de Vida dos Componentes

ngOnChanges()
ngOnInit()
ngDoCheck()
ngAfterContentInit()
ngAfterContentChecked()
ngAfterViewInit()
ngAfterViewChecked()
ngOnDestroy()

Ciclo de Vida dos Componentes

`ngOnChanges()`

- Esse evento é executado uma vez antes do `ngOnInit()`, e toda vez que os valores dos `@Input()` forem alterados
- Se o componente não possui `@Input()`, ou possui porém não faz uso, o `ngOnChanges()` não é executado

Ciclo de Vida dos Componentes

`ngOnInit()`

- Esse evento é executado imediatamente depois que o componente é inicializado
- Ele é executado apenas uma vez, depois do primeiro `ngOnChanges()`
- Ele é executado mesmo que o `ngOnChanges()` nunca seja executado

Ciclo de Vida dos Componentes

`ngDoCheck()`

- Esse evento é executado sempre que o Angular detectar alterações, tanto alterações na interface do usuário como alterações nos dados
- Ele é executado imediatamente depois de cada execução do `ngOnChanges()`, e imediatamente depois do `ngOnInit()`

Ciclo de Vida dos Componentes

`ngAfterContentInit()`

- Esse evento é executado depois que o conteúdo do componente é enviado para a visualização
- Ele é executado apenas uma vez, depois do primeiro `ngDoCheck()`

Ciclo de Vida dos Componentes

`ngAfterContentChecked()`

- Esse evento é executado depois que o Angular realiza uma verificação do conteúdo do componente
- Ele é executado depois do `ngAfterContentInit()`, e depois de cada `ngDoCheck()` subsequente

Ciclo de Vida dos Componentes

`ngAfterViewInit()`

- Esse evento é executado depois que a visualização do componente for totalmente finalizada
- Ele é executado apenas uma vez, depois do primeiro `ngAfterContentChecked()`

Ciclo de Vida dos Componentes

`ngAfterViewChecked()`

- Esse evento é executado depois que o Angular realiza uma verificação da visualização do componente
- Ele é executado depois do `ngAfterViewInit()`, e depois de cada `ngAfterContentChecked()` subsequente

Ciclo de Vida dos Componentes

`ngOnDestroy()`

- Esse evento é executado imediatamente, sempre que o Angular destruir um componente

Data Binding

Data Binding

O data binding cria uma conexão ao vivo entre a interface do usuário do componente e os dados do componente

Ou seja, permite que as variáveis da classe TypeScript sejam exibidas e até controladas pelo html

Data Binding

{{expression}}

- Interpolation
- Permite que valores dinâmicos sejam incorporados no html

Data Binding

```
[target]="expression"
```

- Property Binding
- Permite definir valores dinâmicos para as propriedades dos elementos html

Data Binding

```
(target)="statement"
```

- Event Binding
- Permite associar eventos nos elementos html
- Movimento de mouse, click, pressionamento de teclas, etc

Data Binding

```
[(target)]="expression"
```

- Two-way Binding
- É uma união do Event Binding e do Property Binding
- Permite receber e atualizar valores entre a interface do usuário e a classe TypeScript