**Escopo de Instâncias de Serviços e Módulos no Angular**

No Angular, o escopo de instâncias de serviços e módulos é crucial para entender como as dependências são gerenciadas e como as instâncias de classes são compartilhadas entre diferentes partes da aplicação. Compreender os diferentes escopos permite que você utilize os serviços e módulos de forma eficiente e evite problemas de compartilhamento de dados indesejados.

**1. Escopo de Serviços:**

Os serviços no Angular podem ter três escopos distintos:

- **Root:** Uma única instância do serviço é criada e compartilhada em toda a aplicação.
- **Scoped by Provider:** Uma nova instância do serviço é criada para cada componente ou módulo que o injeta.
- **Scoped by Factory:** Uma instância do serviço é criada dinamicamente com base em lógica personalizada.

**Exemplo 1: Escopo Root:**

TypeScript

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

@Injectable({ providedIn: 'root' })
export class MeuServico {
  // Propriedade e métodos do serviço
}
```

**Uso:**

TypeScript

```typescript
import { Component, OnInit, Inject } from '@angular/core';
import { MeuServico } from './meu-servico';
```

```
@Component({
  selector: 'app-meu-componente',
  template: `
    <p>Valor da propriedade: {{ meuServico.propriedade }}</p>
  `,
})
export class MeuComponente implements OnInit {
  constructor(@Inject(MeuServico) private meuServico: MeuServico) {}

  ngOnInit() {
    // Acessar e modificar propriedades do serviço
  }
}
```

**Exemplo 2: Escopo por Provider:**

TypeScript

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

@Injectable()
export class MeuServico {
  // Propriedade e métodos do serviço
}
```

**Uso:**

TypeScript

```
import { Component, OnInit, Inject } from '@angular/core';
import { MeuServico } from './meu-servico';

@Component({
  selector: 'app-meu-componente',
  template: `
    <p>Valor da propriedade: {{ meuServico.propriedade }}</p>
  `,
  providers: [{ provide: MeuServico, useClass: MeuServico }] // Provider no
componente
})
export class MeuComponente implements OnInit {
  constructor(@Inject(MeuServico) private meuServico: MeuServico) {}

  ngOnInit() {
    // Acessar e modificar propriedades do serviço
  }
```

```
}
```

**Exemplo 3: Escopo por Factory:**

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

@Injectable()
export class MeuServico {
  // Propriedade e métodos do serviço
}

export function meuServicoFactory() {
  // Lógica para criar a instância do serviço
  return new MeuServico();
}
```

**Uso:**

```
TypeScript
import { Component, OnInit, Inject } from '@angular/core';
import { MeuServico } from './meu-servico';

@Component({
  selector: 'app-meu-componente',
  template: `
    <p>Valor da propriedade: {{ meuServico.propriedade }}</p>
  `,
  providers: [{ provide: MeuServico, useFactory: meuServicoFactory }] //
Factory no componente
})
export class MeuComponente implements OnInit {
  constructor(@Inject(MeuServico) private meuServico: MeuServico) {}

  ngOnInit() {
    // Acessar e modificar propriedades do serviço
  }
}
```

**2. Escopo de Módulos:**

Os módulos no Angular definem um escopo para declarações e provedores. Isso significa que os componentes, diretivas e pipes declarados dentro de um módulo só estarão disponíveis para outros componentes dentro do mesmo módulo.

**Exemplo:**

```typescript
import { NgModule } from '@angular/core';
import { MeuComponente } from './meu-componente';
import { MeuServico } from './meu-servico';

@NgModule({
  declarations: [MeuComponente],
  imports: [],
  exports: [MeuComponente],
  providers: [MeuServico] // Servico fornecido no módulo
})
export class MeuModulo {}
```

**Uso:**

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { MeuModulo } from './meu-modulo';

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

**Escopo de Serviços e Módulos no Angular: Detalhes e Boas Práticas**

**3. Gerenciamento Avançado de Escopo:**

O Angular oferece mecanismos mais avançados para gerenciar o escopo de instâncias de serviços e módulos:

- **Lazy Loading:** Carregar módulos e seus serviços associados sob demanda, otimizando o carregamento da aplicação.
- **Injeção de Dependência Hierárquica:** Acessar serviços de módulos ancestrais na hierarquia da árvore de componentes.
- **Singleton:** Garantir que apenas uma instância de um serviço exista em toda a aplicação, mesmo que seja injetado em vários componentes.
- **Multiton:** Permitir que várias instâncias do mesmo serviço existam em diferentes partes da aplicação.

**4. Boas Práticas:**

- **Escolha o Escopo Adequado:** Utilize o escopo correto para cada serviço, considerando a necessidade de compartilhamento de dados e lógica entre os componentes.
- **Evite o Escopo Root Excessivo:** Limite o uso do escopo root para serviços que realmente precisam estar disponíveis em toda a aplicação.
- **Utilize Módulos para Modularidade:** Organize seus serviços e componentes em módulos para melhorar a organização e o reuso de código.
- **Documente o Escopo:** Documente o escopo de cada serviço e módulo para facilitar a compreensão e evitar confusões.

**5. Exemplos Adicionais:**

- **Compartilhando Dados entre Módulos:** Utilize serviços com escopo root ou injeção de dependência hierárquica para compartilhar dados entre módulos.

- **Gerenciando Recursos Externos:** Utilize serviços com escopo root ou singleton para gerenciar recursos externos como conexões com APIs ou arquivos.
- **Criando Instâncias Personalizadas:** Utilize fábricas de serviços para criar instâncias personalizadas de serviços com base em necessidades específicas.

**6. Ferramentas e Recursos:**

- **Documentação Oficial do Angular sobre Escopo de Serviços:** [URL inválido removido]
- **Documentação Oficial do Angular sobre Módulos:** [URL inválido removido]
- **Curso sobre Angular da Loiane Groner:** https://loiane.training/curso/angular

**7. Considerações Finais:**

Compreender o escopo de serviços e módulos é crucial para desenvolver aplicações Angular robustas e escaláveis. Ao escolher o escopo correto e aplicar as boas práticas, você garante que seus serviços sejam utilizados de forma eficiente e que os dados sejam compartilhados de forma controlada.

**8. Exemplos Práticos em Código:**

**Exemplo 1: Serviço com Escopo Root:**

```
TypeScript
// app.module.ts
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { MeuServico } from './meu-servico';
```

```
@NgModule({
  declarations: [AppComponent],
  imports: [],
  bootstrap: [AppComponent],
  providers: [MeuServico] // Serviço fornecido no módulo root
})
export class AppModule {}
```

**Exemplo 2: Serviço com Escopo por Provider:**

TypeScript

```
// app.component.ts
import { Component, OnInit, Inject } from '@angular/core';
import { MeuServico } from './meu-servico';

@Component({
  selector: 'app-root',
  template: `
    <p>Valor da propriedade: {{ meuServico.propriedade }}</p>
  `,
  providers: [{ provide: MeuServico, useClass: MeuServico }] // Provider no
componente
})
export class AppComponent implements OnInit {
  constructor(@Inject(MeuServico) private meuServico: MeuServico) {}

  ngOnInit() {
    // Acessar e modificar propriedades do serviço
  }
}
```

**Exemplo 3: Serviço com Escopo por Factory:**

TypeScript

```
// meu-servico.factory.ts
import { Injectable } from '@angular/core';
import { MeuServico } from './meu-servico';

export function meuServicoFactory() {
  // Lógica para criar a instância do serviço
  return new MeuServico();
}
```

**Exemplo 4: Módulo com Declarações e Provedores:**

TypeScript

```typescript
// meu-modulo.ts
import { NgModule } from '@angular/core';
import { MeuComponente } from './meu-componente';
import { MeuServico } from './meu-servico';

@NgModule({
  declarations: [MeuComponente],
  imports: [],
  exports: [MeuComponente],
  providers: [MeuServico] // Serviço fornecido no módulo
})
export class MeuModulo {}
```