ARS

Motivation

Ausgangsbasis

- Geschäftslogik in Komponentenklassen
- Manuelle Initialisierung von Abhängigkeiten

Wunsch

- Techniken und Konzepte für die Strukturierung von Geschäftslogik
- Wiederverwendung ermöglichen (DRY)
- Wartbarkeit sicherstellen durch Modularisierung
- Abhängigkeitsverwaltung durch das Framework

ARS

Lernziele

- Sie erklären die Bedeutung und den Nutzen von Services.
- Sie beschreiben was Dependency Injection ist und wie diese in Angular umgesetzt ist.
- Sie erklären den Unterschied zwischen Services und Komponenten.

ARS

Abgrenzung zwischen Komponenten und Services

- **Komponente**:
  HTML-Template + Komponentenklasse für die View-Logik
- **Service:**
  Einfache TypeScript-Klasse mit Methoden und Objektvariablen
  - Business-Logik
  - Funktionalität, die mehrere Komponenten nutzen können/sollen
  - Daten verarbeiten (sichern und laden)
  - Beispiel: Zugriff auf ein Back-End

https://angular.io/generated/images/guide/architecture/service.png

ARS

Komponenten und Services – Einblick in Angular Way

- "Component classes should be lean. They don't fetch data from the server, validate user input, or log directly to the console. They delegate such tasks to services."

- "A component's job is to enable the user experience and nothing more. It mediates between the view (rendered by the template) and the application logic (which often includes some notion of a model). A good component presents properties and methods for data binding. It delegates everything nontrivial to services."

https://angular.io/guide/architecture#services

ARS

Service – Eigenschaften

- Services sind Singletons
- Jeder Service ist global verfügbar
- Komponenten können auf Services zugreifen
- Service Instanz wird per Dependency Injection in Komponente injiziert
- Konfiguration der Dependency Injection
  - Service-Klasse mit @Injectable Dekorator markieren
    → Kommuniziert an Angular, was inijzierbar ist

    → providedIn Eigenschaft des Dekorators legt fest in welchem Modul der Service verfügbar ist
  - Definition eines Parameters im Konstruktor mit Angabe des Typs kommuniziert an Angular, was injiziert werden soll

ARS

Implementierung eines Services – Beispiel

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

@Injectable({providedIn: 'root'})
class ShoppingCartService {
    private products = [];

    public add(product: Product) {
        this.products.push(product)
    }

    public getAll(): Product[] {
        return this.products;
    }
}
```

Service wird in ‚root' registriert

ARS

Verwenden eines Services – Beispiel

```
import { ShoppingCartService } from 'shopping-cart.service';

@Component({ … })
class ShoppingCartComponent  implements OnInit {
   private products: Product[];

   constuctor
      (private shoppingCartService: ShoppingCartService) {}

   ngOnInit() {
       this.products = this.shoppingCartService.getAll();
   }
}
```

ARS

Zusammenfassung

Um was ging es in diesem Modul?

- Grundlegender Baustein von Angular: Services
- Unterscheid zwischen Services und Komponenten
- Dependency Injection in Angular

Wozu brauche ich das? Was will ich damit machen?

- Zentrale Steuerung von Abhängigkeiten
- Wiederverwendbarkeit durch Kapselung von Geschäftslogik in Services
- Wartbarkeit sicherstellen durch Strukturierung Services

ARS

Kontrollfragen

- Was ist der Unterscheid zwischen Services und Komponenten?
- Wann sollte ein Service, wann eine Komponente benutzt werden
- Wie werden Services im Dependency Injection System von Angular angemeldet?