ANGULAR FRAMEWORK ile WEB GELİŞTİRME

Şevval Yavuz
21360859031
Bilgisayar Mühendisliği-3.Sınıf

09.05.2024

İçindekiler

Angular Nedir?

- **Google tarafından geliştirilmiş, açık kaynaklı bir JavaScript frameworküdür.**
- **SPA (Single Page Application)**
- **Web,mobil ve masaüstü uygulamalar için kullanılır.**
- **TypeScript ile birlikte kullanılır.**

Angular ın Temelleri

- **Component**
- **Servisler(Services)**
- **Modüller(Modules)**
- **Veri Bağlama (Data Binding)**
- **Rotalama**
- **Borular (Pipes)**
- **Formlar(Forms)**
- **Http İstekleri**

Component

Angular uygulamalarındaki temel yapı taşlarıdır.

Her component birlikte çalışan iki ana parçadan oluşur.

- **HTML Templatı**
- **TypeScript Sınıfı**

Component

HTML Templatı ve TypeScript Sınıfı

component.html

```
<h1>Merhaba, {{ isim }}!</h1>
```

component.ts

```
export class
MerhabaComponent {
isim = 'Dünya';
}
```

Component

Faydaları

- **Kod Tekrarlanabilirliği**
- **Modülerlik**

Servisler(Services)

Angular uygulamasının farklı bileşenleri arasında veri paylaşımı ve işlevselliği yeniden kullanmayı sağlayan sınıflardır.

Bir Angular servisi, aşağıdaki bileşenlerden oluşur.

- **@Injectible Dekoratörü**
- **Consturctor**
- **Metodlar**
- **Değişkenler**

Servisler(Services)

kullanici.service.ts

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

@Injectable({
 providedIn: 'root'
})
export class KullaniciService {
 private kullanicilar: any[] = [];

 constructor(private http: HttpClient) { }

 getKullanicilar() {
  return this.http.get<any[]>
('https://api.example.com/kullanicilar');
 }

 kullaniciEkle(kullanici: any) {
  // Kullanıcı ekleme işlemini gerçekleştirin
 }

}
```

Servisler(Services)

Faydaları

- **Kod Tekrarlanabilirliği**

Aynı kodu bir componentte tekrar tekrar yazmayız.

- **Kod Organizasyonu**

Okunabilirlik sağlar.

- **Test Edilebilirlik**

Servisler, diğer componentlerden ve servislerden bağımsız olarak çalışabilir.

Modüller(Modules)

Angular bir modülden oluşur.

Modüller bileşenleri tek bir yerde toplayan ve bunların etkileşimini sağlayan ana mimaridir.

Modüller(Modules)

Bir Angular modülü,genellikle aşağıdaki bileşenlerden oluşur:

- **@NgModule Dekoratörü**
- **İçeri Alınan Bileşenler (Components)**
- **Diğer Modüller**

Aynı zamanda servisleri,boruları ve yönergeleri de modüle dahil edebilirsiniz.

Modüller(Modules)

Basit bir kullanıcı modülü oluşturabilirsiniz.

module.ts

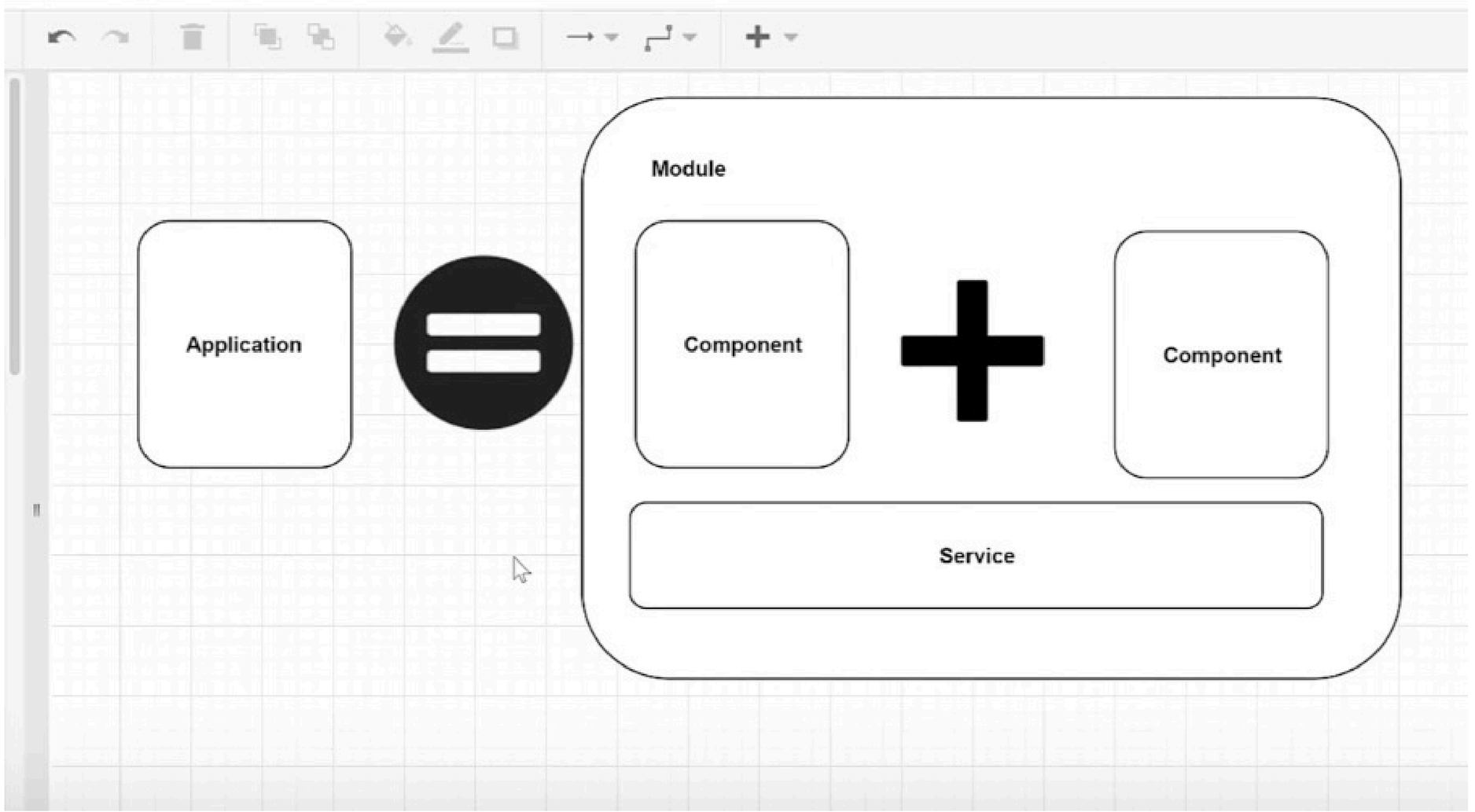
```
import { NgModule } from
'@angular/core';
import { KullaniciListComponent } from
'./kullanici-list.component';
import { KullaniciDetayComponent }
from './kullanici-detay.component';
import { KullaniciService } from
'./kullanici.service';

@NgModule({
 declarations: [KullaniciListComponent,
KullaniciDetayComponent],
 imports: [],
 providers: [KullaniciService]
})
export class KullaniciModule { }
```

Modüller(Modules)

Faydaları

- **Kod Organizasyonu**
- **Kod Tekrarlanabilirliği**
- **Bağımlılık Yönetimi**
- **Tembel Yükleme(Lazy Loading)**


Application
Module
Component
Component
Service

Veri Bağlama (Data Binding)

Veri bağlama, componentlerinizdeki veriler ile HTML templatınızdaki öğeler arasında iki yönlü bir senkronizasyon sağlar.

İki temel yolu vardır.

- **Property Binding(Özellik Bağlama)**
- **Event Binding (Olay Bağlama)**

Ayrıca İki Yönlü Bağlama (Two-Way Binding) bulunur.

Veri Bağlama (Data Binding)

1)Property Binding

Örneğin:

```
<h1 id="baslik">{{ baslikMetni }}</h1>//
component.ts
export class OrnekComponent {
baslikMetni = 'Merhaba Angular!';
}
```

Veri Bağlama (Data Binding)

1)Event Binding

Örneğin:

```
<button (click)="sayaciArttir()">Tıkla</button>

// component.ts
export class OrnekComponent {
 sayac = 0;

 sayaciArttir() {
  this.sayac++;
 }
}
```

Veri Bağlama (Data Binding)

Two Way Binding

Angular ayrıca *ngModel* direktifi ile iki yönlü veri bağlamayı sağlar.

Örnek:

```
<input type="text" [(ngModel)]="kullaniciAdi">

// component.ts
export class OrnekComponent {
 kullaniciAdi = '';
}
```

Veri Bağlama (Data Binding)

Faydaları

- **Kod tekrarlanabilirliğini azaltır.**
- **Kodun okunabilirliğini ve bakım kolaylılığını arttırır.**
- **Dinamik ve etkileşimli**

Rotalama(Routing)

Angular rotalama, URL’deki değişikliklere göre hangi componentin gösterileceğini belirleyen sistemdir.

Rotalar genellikle *AppRoutingModule* isimli modülde tanımlanır.

Bu modül *RouterModule.forRoot* metodu ile uygulamaya eklenir.

Rotalama(Routing)

Rota Tanımlama Örneği

```
TypeScript
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { HakkimizdaComponent } from
'./hakkimizda/hakkimizda.component';
import { KullanicilarComponent } from
'./kullanicilar/kullanicilar.component';

const routes: Routes = [
 { path: '', component: HomeComponent },
 { path: 'hakkimizda', component: HakkimizdaComponent },
 { path: 'kullanicilar', component: KullanicilarComponent },
];

@NgModule({
 imports: [RouterModule.forRoot(routes)],
 exports: [RouterModule]
})
export class AppRoutingModule { }
```

Rotalama(Routing)

Faydaları

- **Tek sayfa uygulamalarda sayfa geçişlerini kolaylaştırır.**
- **Uygulamayı modüler ve organize tutar.**
- **Derin linkleme (deep linking) imkanı sağlar.**

Borular(Pipes)

Angular boruları componentlerinizdeki verileri biçimlendirmek ve görüntülemek için kullandığınız dönüştürme işlemleridir.

Pipes componentinizdeki şablonlarda kullanılır.

Örnek:

```
<p>Ürün fiyatı: {{ fiyat | currency:'TRY' }}</p>

// component.ts
export class UrunComponent {
 fiyat = 123.45;
}
```

Borular(Pipes)

Faydaları

- **Kod Tekrarlanabilirliği**
- **Bakım Kolaylığı**
- **Okunabilirlik**

Formlar

Angular formları kullanıcıların uygulama ile etkileşime girmesi için önemli bir etkendir.

Angular iki tür form yaklaşımı sunar.

- **Template-Driven Forms (Şablon Yönetimli Formlar)**
- **Reactive Forms (Reaktif Formlar)**

Formlar

1)Template-Driven Forms

Avantajları:

- **Kurulumu ve kullanımı daha kolaydır.**
- **Basit formlar için idealdir.**

Dezavantajları:

- **Karmaşık formlarda kod okunabilirliği azalabilir.**
- **Daha az esneklik sunar.**

Formlar

2)Reactive Forms

Avantajları:

- **Daha fazla esneklik ve denetim imkanı sunar.**
- **Daha karmaşık formlar için idealdir.**

Dezavantajları:

- **Kurulumu ve kullanımı template-driven a göre karmaşıktır**

Formlar

Faydaları

- **Kullanıcı girişi yönetimi**
- **Form doğrulama**
- **Form verilerinin işlenmesi**

HTTP İstekleri

Angular uygulamalarında sunucudan veri almak veya sunucuya veri göndermek için HTTP istekleri kullanılır.

Bunu yaparken HttpClient sınıfını kullanır.

HTTP İstekleri

HttpClient Kullanımı

1)Import

@angular/common/http

2)Enjeksiyon

Componente enjekte etme

3)İstek Yöntemleri

get(url),post(url,data),put(url,data),delete(url)

4)İstek Gönderme

Seçtiğimiz istek yöntemini kullanarak istek göndeririz.

5)Tepki Alma

subscribe metodu ile gelen yanıtı yakalayabiliriz.

Angular hakkında daha fazla bilgi için:
Angular Resmi Web Sitesi: https://angular.io/

Kaynakça

https://www.btkakademi.gov.tr/portal/course/player/deliver/angular-7-9914

https://www.youtube.com/@sadik_turan

https://github.com/AEDD1970/pruebaKuepa

https://stackoverflow.com/questions/71786214/why-bootstrap-key-in-ngmodule-is-not-working-when-you-build-the-same-structur

https://github.com/ert550606/angularfinal

Sorularınız?

Beni dinlediğiniz için teşekkür ederim.