Understanding Angular Modules: A Beginner's Guide

1 What is an Angular Module?

In Angular, a module is a container for a cohesive block of code dedicated to an application domain, a workflow, or a set of related capabilities. Modules help in organizing an Angular application into smaller, manageable pieces. They group components, directives, pipes, and services into cohesive blocks.

2 Key Concepts

- **Root Module**: Every Angular application has at least one module, the root module, typically named `AppModule`. It bootstraps the application and is the entry point of your application.
- **Feature Modules**: These are modules that help you organize your application into different features. For instance, you might have a `UserModule` for user-related components and services.
- **Shared Modules**: These modules contain components, directives, and pipes that are used across multiple other modules.
- **Lazy-loaded Modules**: These modules are loaded on-demand. This helps in improving the application's performance by loading only the necessary code.

3 Structure of a Module

An Angular module is defined using the `@NgModule` decorator and typically includes:

- `declarations`: Components, directives, and pipes that belong to the module.
- `imports`: Other modules whose exported components, directives, or pipes are needed by the components in this module.

- `exports`: Components, directives, and pipes that can be used by other modules.
- `providers`: Services that are available to the components in this module.
- `bootstrap`: The main application view, called the root component, which hosts all other app views.

4 Example: Two Entities with a Many-to-One Relationship

Let's say you are building a simple application with two entities: `Order` and `Customer`. Each `Order` is related to a single `Customer`, but a `Customer` can have multiple `Orders`.

4.1 Step 1: Create the Modules

```
ng generate module order
ng generate component order-list
ng generate component order-detail

ng generate module customer
ng generate component customer-list
ng generate component customer-detail
```

4.2 Step 2: Define the Modules

Listing 1: Order Module (order.module.ts)

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { OrderListComponent } from './order-list/order-list.component';
import { OrderDetailComponent } from './order-detail/order-detail.component'

@NgModule({
  declarations: [
    OrderListComponent,
    OrderDetailComponent
  ],
  imports: [
    CommonModule
  ],
  exports: [
    OrderListComponent,
```

```
    OrderDetailComponent
  ]
})
export class OrderModule { }
```

Listing 2: Customer Module (customer.module.ts)

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { CustomerListComponent } from './customer-list/customer-list.componer
import { CustomerDetailComponent } from './customer-detail/customer-detail.c

@NgModule({
  declarations: [
    CustomerListComponent,
    CustomerDetailComponent
  ],
  imports: [
    CommonModule
  ],
  exports: [
    CustomerListComponent,
    CustomerDetailComponent
  ]
})
export class CustomerModule { }
```

4.3 Step 3: Use the Modules in the Root Module (app.module.ts)

Listing 3: Root Module (app.module.ts)

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    OrderModule,
    CustomerModule
```

```
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

4.4 Step 4: Set Up Relationships in Components

Listing 4: Order Component Example (order-list.component.ts)

```
import { Component, OnInit } from '@angular/core';
import { OrderService } from '../order.service';
import { CustomerService } from '../customer.service';

@Component({
  selector: 'app-order-list',
  templateUrl: './order-list.component.html'
})
export class OrderListComponent implements OnInit {
  orders = [];
  customers = [];

  constructor(private orderService: OrderService, private customerService: C

  ngOnInit() {
    this.orders = this.orderService.getOrders();
    this.customers = this.customerService.getCustomers();
  }
}
```

Listing 5: Customer Component Example (customer-detail.component.ts)

```
import { Component, OnInit, Input } from '@angular/core';
import { Customer } from '../customer';
import { OrderService } from '../order.service';

@Component({
  selector: 'app-customer-detail',
  templateUrl: './customer-detail.component.html'
})
export class CustomerDetailComponent implements OnInit {
  @Input() customer: Customer;
  orders = [];

  constructor(private orderService: OrderService) {}
```

```
14
15    ngOnInit() {
16      this.orders = this.orderService.getOrdersByCustomer(this.customer.id);
17    }
18  }
```

5 Summary

In this guide, you learned the basics of Angular modules, including their purpose and structure. You also saw how to create and organize modules for entities with a many-to-one relationship, such as `Order` and `Customer`. Modules help in managing and scaling your Angular application effectively.