Child Routes

In Angular, child routes allow you to create nested routes within a feature module. Child routes are configured within the `children` property of a route configuration. This enables you to organize your application into modular and reusable components. Child routes are often used when you have a parent component that needs to load and display different views based on user interaction or navigation.

Here is an example to illustrate the concept of child routes:

Step 1: Define Parent and Child Components

Create the parent and child components:

```
ng generate component parent
ng generate component parent/child
```

Step 2: Configure Routes

In the `app-routing.module.ts`, configure the parent and child routes:

```
// app-routing.module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { ParentComponent } from './parent/parent.component';
import { ChildComponent } from './parent/child/child.component';

const routes: Routes = [
  { path: 'parent', component: ParentComponent, children: [
    { path: 'child', component: ChildComponent },
    // Additional child routes can be added here
  ]},
  // Other routes...
];

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

Step 3: Configure Parent Component

In the `parent.component.html`, add a `<router-outlet>` where the child components will be displayed:

```
<!-- parent.component.html -->
<h2>Parent Component</h2>
<nav>
  <a routerLink="/parent">Parent</a>
  <a routerLink="/parent/child">Child</a>
</nav>

<router-outlet></router-outlet>
```

Step 4: Configure Child Component

In the `child.component.html`, add content for the child component:

```
<!-- child.component.html -->
<h3>Child Component</h3>
<p>This is the content of the child component.</p>
```

Step 5: Run Your App

Run your Angular application:

```
ng serve
```

Navigate to `http://localhost:4200/parent` and `http://localhost:4200/parent/child` in your browser. You should see the content of the parent and child components respectively.

Notes:

- The `children` property is used in the route configuration to define child routes.
- The `<router-outlet>` directive in the parent component's template is a placeholder where the child components will be dynamically inserted.
- Child routes can have their own child routes, creating a hierarchy of routes.
- Each level of the route hierarchy corresponds to a component hierarchy, making it easier to organize and manage complex applications.

Child routes provide a powerful way to structure your Angular application, especially when dealing with larger applications that require modularization and organization.

Let's create a sample Angular application with the structure you've provided using Angular Router.

Step 1: Generate Components

Generate the components for `OnlineApplication`, `Appliances`, `Furniture`, `Lighting`, `Electronics`, `Mobiles`, `Laptop`, `Fashion`, `Men`, `Women`, and `Login`:

```
ng generate component OnlineApplication
ng generate component Appliances
ng generate component Furniture
ng generate component Lighting
ng generate component Electronics
ng generate component Mobiles
ng generate component Laptop
ng generate component Fashion
ng generate component Men
ng generate component Women
ng generate component Login
```

Step 2: Configure Routes

In the `app-routing.module.ts`, configure the routes for the components:

```
// app-routing.module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { OnlineApplicationComponent } from './online-application/online-
application.component';
import { AppliancesComponent } from './appliances/appliances.component';
import { FurnitureComponent } from './furniture/furniture.component';
import { LightingComponent } from './lighting/lighting.component';
import { ElectronicsComponent } from './electronics/electronics.component';
import { MobilesComponent } from './mobiles/mobiles.component';
import { LaptopComponent } from './laptop/laptop.component';
import { FashionComponent } from './fashion/fashion.component';
import { MenComponent } from './men/men.component';
import { WomenComponent } from './women/women.component';
import { LoginComponent } from './login/login.component';

const routes: Routes = [
  { path: 'online-application', component: OnlineApplicationComponent,
children: [
    { path: 'appliances', component: AppliancesComponent, children: [
      { path: 'furniture', component: FurnitureComponent },
      { path: 'lighting', component: LightingComponent }
    ]},
    { path: 'electronics', component: ElectronicsComponent, children: [
      { path: 'mobiles', component: MobilesComponent },
      { path: 'laptop', component: LaptopComponent }
    ]},
    { path: 'fashion', component: FashionComponent, children: [
```

```
      { path: 'men', component: MenComponent },
      { path: 'women', component: WomenComponent }
    ]},
    { path: 'login', component: LoginComponent }
  ]},
  // Other routes...
];

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

Step 3: Configure Components

Update the HTML templates of each component to add navigation links. For example, in `online-application.component.html`:

```
<!-- online-application.component.html -->
<h2>Online Application</h2>
<nav>
  <a routerLink="/online-application/appliances">Appliances</a>
  <a routerLink="/online-application/electronics">Electronics</a>
  <a routerLink="/online-application/fashion">Fashion</a>
  <a routerLink="/online-application/login">Login</a>
</nav>

<router-outlet></router-outlet>
```

Repeat this pattern for other components.

Step 4: Run Your App

Run your Angular application:

```
ng serve
```

Navigate to `http://localhost:4200/online-application`, `http://localhost:4200/online-application/appliances`, etc., to see the components and child components.

This example demonstrates a simple hierarchical structure using Angular Router for the provided components. Adjust the content and styles of each component as needed for your application.