混合应用开发

--- Angular路由

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- **路由配置**
- **路由传参**
- **路由JS跳转**
- **父子路由**

路由

- 路由（router）
  - 应用程序一般包含多个视图 (view)。 用户通过点击链接、按钮等，在视图之间导航
  - Angular 中的路由是一个特性丰富的机制，可以配置和管理整个导航过程，包括建立和销毁视图
- 路由配置
  - index.html的<head>标签下添加<base>元素，告诉路由该如何合成导航用的URL

路由

- 路由配置（续）
  - 引入 RouterModule 模块
    - import { RouterModule } from '@angular/router';
  - 通过 RouterModule.forRoot 方法配置路由信息(该方法位于imports数组中)
    - RouterModule.forRoot([

      { path: 'home', component: HomeComponent }, ...

      ])

路由

- 路由配置（续）
  - 路由出口 -- <router-outlet></router-outlet>
    - 显示路由生成的视图的元素
  - 路由链接 -- <a [routerLink]= [“/”] [routerLinkActive]=[“”] >
    - routerLink：绑定导航的路径
    - routerLinkActive：将CSS类添加到激活路由的元素上

路由

- 重定向路由（默认路由）
  - RouterModule.forRoot([
    { path: “”，redirectTo:“默认路由路径”，pathMatch:“full”}
    ])
- 通配符路由
  - RouterModule.forRoot([
    { path: “**”，component:“组件名”}
    ])

路由守卫

- 路由守卫
  - CanActivate：处理导航到某路由的情况
  - CanDeactivate：处理从当前路由离开的情况
  - Resolve：在路由激活之前获取路由数据

路由守卫

- CanActivate 守卫
  - 创建守卫类
    - export class GuardClass implements CanActivate
  - 配置路由
    - import { GuardClass } from ‘’ ;
    - {path: ‘ ',...,canActivate:[GuardClass]}
  - 注入服务
    - providers:[GuardClass]

路由守卫

- CanDeactivate 守卫
  - 创建守卫类
    - export class GuardClass implements CanDeactivate<Component>{ }
  - 配置路由
    - import { GuardClass } from ‘ ’ ;
    - {path: ‘ ',..., canDeactivate:[GuardClass]}
  - 注入服务
    - providers:[GuardClass]

路由守卫

- Resolve 守卫
  - 创建守卫类
    - export class GuardClass implements CanDeactivate<Component>{ }
  - 配置路由
    - import { GuardClass } from ‘ ’ ;
    - {path: ‘ ',..., canDeactivate:[GuardClass]}
  - 注入服务
    - providers:[GuardClass]

内容提纲

- 路由配置
- 路由传参
- 路由JS跳转
- 父子路由

路由传参

- 路由路径中传递
  - 在路由定义中声明参数
    - RouterModule.forRoot([

      { path: “home/：id”, component:“组件名” }

      ])
  - 在实际路径中携带参数
    - <a [routerLink]= [“/”,参数值] >

路由传参（续）

- 在目标组件中接收数据
  - import { ActivatedRoute } from '@angular/router';
  - constructor（private route: ActivatedRoute）{ }
  - this.route.snapshot.params[“param”]

- 查询参数中传递
  - <a [routerLink]= [“/”] [queryParams]= “{属性：属性值}” >
  - 在目标组件中接收数据
    - this.route.snapshot.queryParams[“param”]

数据接收

- 数据接收
  - 参数快照
    - 如果视图切换时切换到同一组件，参数快照形式参数不变
    - this.route.snapshot.queryParams[“param”];
  - 参数订阅
    - 如果视图切换时切换到同一组件，参数依然会随之变化
    - this.route.params.subscribe(params=>{ });

内容提纲

- 路由配置
- 路由传参
- 路由JS跳转
- 父子路由

JS跳转

- JS跳转
  - 引入 Router 模块
    - import { Router } from '@angular/router';
  - 构造函数中声明
    - constructor（public _router：Router）{ }
  - 调用 Router.navigate ( ) 方法
    - 路径传参时：navigate（[“” ,参数值]）
    - 查询参数传参时：navigate([“”],{ queryParams:{ } });

内容提纲

- 路由配置
- 路由传参
- 路由JS跳转
- 父子路由

父子路由

- Angular允许一个路由组件被嵌入另一个路由组件中，建立路由的多级嵌套关系
- 父子路由
  - { path : “路径”, component : “组件名”, children :[
    { path: “路径”, component: “组件名”}, ……
    ]}

Thank You

河北师范大学软件学院
Software College of Hebei Normal University