Angular06

作业1:

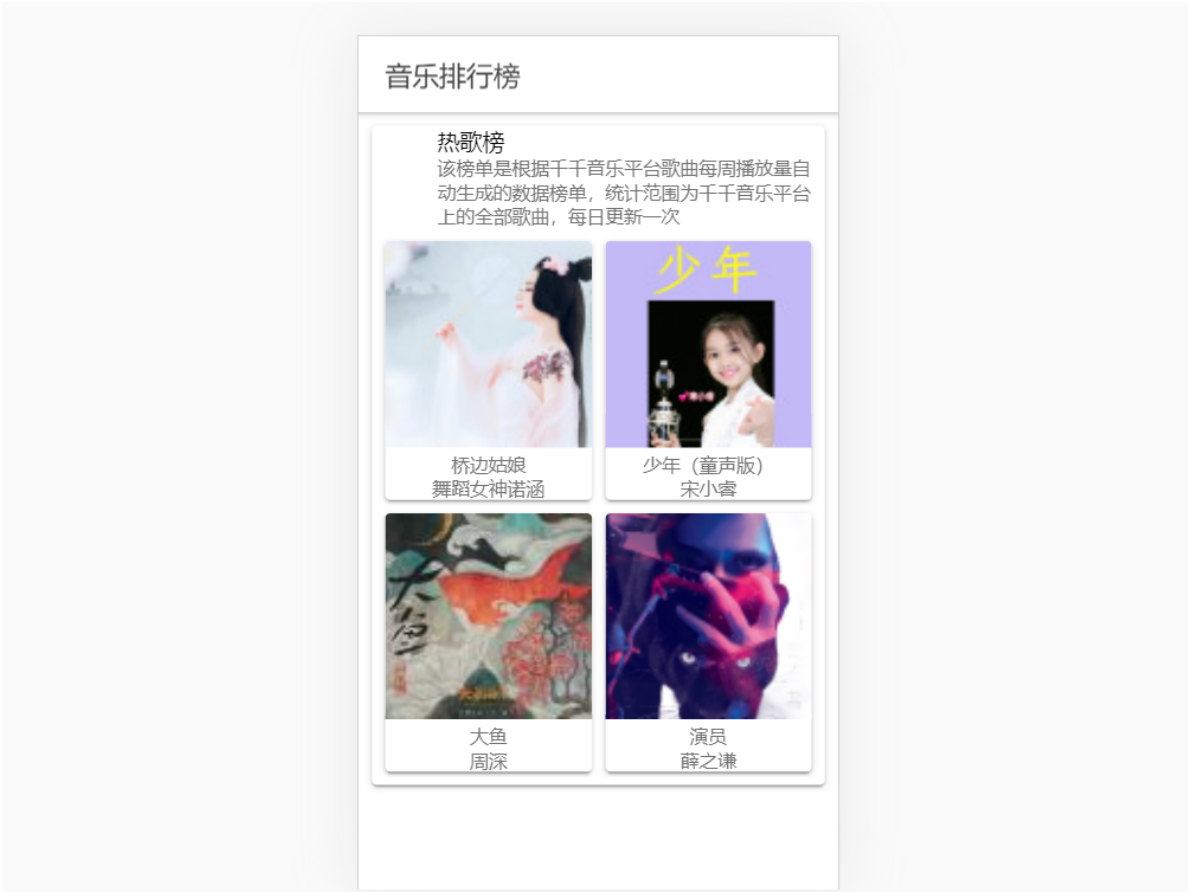

```
import { Component } from "@angular/core";

import { Platform } from "@ionic/angular";
import { SplashScreen } from "@ionic-native/splash-screen/ngx";
import { StatusBar } from "@ionic-native/status-bar/ngx";
import { HttpClient } from "@angular/common/http";

@Component({
  selector: "app-root",
  templateUrl: "app.component.html",
  styleUrls: ["app.component.scss"],
})
export class AppComponent {
  // 考虑到此项目要下拉刷新, 加载更多, 首次 都要使用网络请求:  所以地址会复用.
  // 面向对象的属性:  具有复用性
  url = "https://api.apiopen.top/getJoke?count=5&type=gif&page=";

  //网络请求需要网络服务, 网络服务是网络模块中的: 网络模块不加载 app.module.ts

  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar,
    private http: HttpClient
  ) {
    this.initializeApp();
  }
```

```
  initializeApp() {
    this.platform.ready().then(() => {
      this.statusBar.styleDefault();
      this.splashScreen.hide();
    });
  }

  res: Result;

  ngOnInit(): void {
    // 组件初始化
    this.http.get(this.url + 1).subscribe((res: Result) => {
      console.log(res);

      this.res = res; // 局部变量 保存给属性. 因为属性才能在html中使用
    });
  }

  pno = 1; //当前页

  loadData(e) {
    let nextP = this.pno + 1;

    this.http.get(this.url + nextP).subscribe((res: Result) => {
      console.log(res);

      res.result = this.res.result.concat(res.result);
      this.res = res;

      this.pno = nextP;
      e.target.complete(); //完成加载更多
    });
  }

  doRefresh(e) {
    this.http.get(this.url + 1).subscribe((res: Result) => {
      // 覆盖数据->重置页数->结束动画
      this.res = res;
      this.pno = 1;
      e.target.complete();
    });
  }
}

// 类型声明
interface Result {
  code: number;
  message: string;
  result: ResultData[];
}

interface ResultData {
  comment: string;
  down: string;
  forward: string;
  header: string;
  images: string;
  name: string;
```

```
  passtime: string;
  sid: string;
  text: string;
  thumbnail: string;
  top_comments_content: string;
  top_comments_header: string;
  top_comments_name: string;
  top_comments_uid: string;
  top_comments_voiceuri: string;
  type: string;
  uid: string;
  up: string;
  video: string;
}
```

```
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>段子列表</ion-title>
    </ion-toolbar>
  </ion-header>

  <!-- 使用网络数据 -->
  <ion-content *ngIf="res">
    <ion-refresher slot="fixed" (ionRefresh)="doRefresh($event)">
      <ion-refresher-content></ion-refresher-content>
    </ion-refresher>

    <ion-card *ngFor="let item of res.result">
      <div style="display: flex;">
        <img [src]="item.header" alt="" style="width: 50px; height: 50px;" />
        <div style="display: flex; flex-direction: column; margin-left: 10px;">
          <span style="color: black; font-size: 1.2em;">{{item.name}}</span>
          <span style="margin-top: auto;">{{item.passtime}}</span>
        </div>
      </div>
      <div style="padding: 10px; background-color: lightgray; color: black;">
        {{item.text}}
      </div>
      <img [src]="item.images" alt="" width="100%" />
    </ion-card>

    <ion-infinite-scroll
      threshold="25%"
      position="bottom"
      (ionInfinite)="loadData($event)"
    >
      <ion-infinite-scroll-content
        loadingSpinner="bubbles"
        loadingText="加载中..."
      >
      </ion-infinite-scroll-content>
    </ion-infinite-scroll>
  </ion-content>
</ion-app>
```

作业2



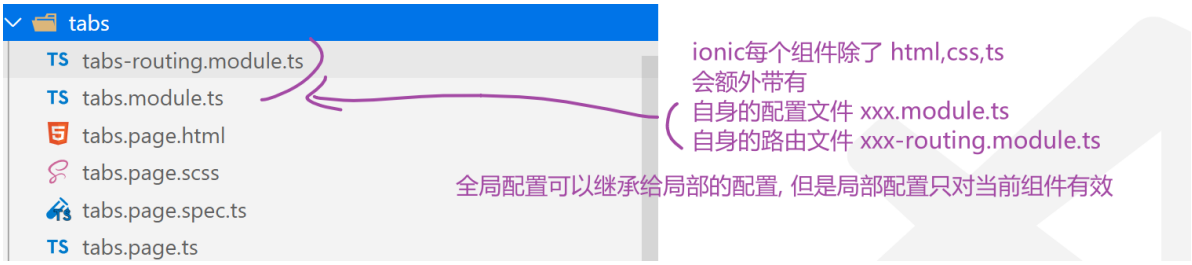

```
import { Component } from "@angular/core";

import { Platform } from "@ionic/angular";
import { SplashScreen } from "@ionic-native/splash-screen/ngx";
import { StatusBar } from "@ionic-native/status-bar/ngx";
import { HttpClient } from "@angular/common/http";

@Component({
  selector: "app-root",
  templateUrl: "app.component.html",
  styleUrls: ["app.component.scss"],
})
export class AppComponent {
  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar,
    private http: HttpClient
  ) {
    this.initializeApp();
  }

  initializeApp() {
    this.platform.ready().then(() => {
      this.statusBar.styleDefault();
      this.splashScreen.hide();
    });
  }
```

```
  res: Result;

  ngOnInit(): void {
    let url = "https://api.apiopen.top/musicRankings";

    this.http.get(url).subscribe((res: Result) => {
      console.log(res);

      this.res = res;
    });
  }
}

interface Result {
  code: number;
  message: string;
  result: ResultData[];
}

interface ResultData {
  bg_color: string;
  bg_pic: string;
  color: string;
  comment: string;
  content: Content[];
  count: number;
  name: string;
  pic_s192: string;
  pic_s210: string;
  pic_s260: string;
  pic_s444: string;
  type: number;
  web_url: string;
}

interface Content {
  album_id: string;
  album_title: string;
  all_rate: string;
  author: string;
  biaoshi: string;
  pic_big: string;
  pic_small: string;
  rank_change: string;
  song_id: string;
  title: string;
}
```

```
<ion-app>
  <ion-header>
    <ion-toolbar>
      <ion-title>音乐排行榜</ion-title>
    </ion-toolbar>
  </ion-header>
```

```
  <ion-content *ngIf="res">
    <ion-slides>
      <ion-slide *ngFor="let item of res.result">
        <ion-card>
          <div style="display: flex;">
            <img [src]="item.pic_s192" alt="" style="width: 100px;" />
            <div
              style="
                display: flex;
                flex-direction: column;
                align-items: flex-start;
                margin-left: 10px;
              "
            >
              <span style="color: black; font-size: 1.2em;">{{item.name}}</span>
              <span style="text-align: start; margin-top: auto;"
                >{{item.comment}}</span
              >
            </div>
          </div>

          <ion-grid fixed>
            <ion-row>
              <ion-col size="6" *ngFor="let content of item.content">
                <ion-card style="margin: 0;">
                  <img [src]="content.pic_small" alt="" style="width: 100%;" />
                  <div>{{content.title}}</div>
                  <div>{{content.author}}</div>
                </ion-card>
              </ion-col>
            </ion-row>
          </ion-grid>
        </ion-card>
      </ion-slide>
    </ion-slides>
  </ion-content>
</ion-app>
```

路由系统



为 tabs 标签栏 新增一个页面:

```
生成页面命令
ionic g page 页面名
```

此处声明tab4: `ionic g page tab4`

```
D:\WEBTN2005\18_Angular\Day06\tabsApp>ionic g page tab4
> ng.cmd generate page tab4 --project=app ionic本质还是 ng命令，即此处的命令
CREATE src/app/tab4/tab4-routing.module.ts (339 bytes) 当前文件的子路由配置
CREATE src/app/tab4/tab4.module.ts (458 bytes) 当前文件的配置
CREATE src/app/tab4/tab4.page.html (123 bytes)
CREATE src/app/tab4/tab4.page.spec.ts (633 bytes)
CREATE src/app/tab4/tab4.page.ts (248 bytes)
CREATE src/app/tab4/tab4.page.scss (0 bytes)
UPDATE src/app/app-routing.module.ts (526 bytes) 生成的页面会自动配置到 总路由文件
[OK] Generated page!
```

组件代码，spec.ts是测试文件,对开发没用

生成页面默认会添加到 根路由文件中

带有 tabs 栏目的页面，必须放在 tabs 路由下.

src/app/tabs/tabs-routing.module.ts

```
import { NgModule } from "@angular/core";
import { RouterModule, Routes } from "@angular/router";
import { TabsPage } from "./tabs.page";

const routes: Routes = [
  {
    path: "tabs",
    component: TabsPage,
    children: [
      {
        path: "tab1",
        loadChildren: () =>
          import("../tab1/tab1.module").then((m) => m.Tab1PageModule),
      },
      {
        path: "tab2",
        loadChildren: () =>
          import("../tab2/tab2.module").then((m) => m.Tab2PageModule),
      },
      {
        path: "tab3",
        loadChildren: () =>
          import("../tab3/tab3.module").then((m) => m.Tab3PageModule),
      },
      {
        path: "tab4",
        loadChildren: () =>
          import("../tab4/tab4.module").then((m) => m.Tab4PageModule),
      },
      {
        path: "",
        redirectTo: "/tabs/tab1",
        pathMatch: "full",
      },
    ],
  },
  {
    path: "",
    redirectTo: "/tabs/tab1",
    pathMatch: "full",
  },
```

```
];

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

tabs栏的UI配置

```
src/app/tabs/tabs.page.html
```

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon name="triangle"></ion-icon>
      <ion-label>Tab 1</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon name="ellipse"></ion-icon>
      <ion-label>Tab 2</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab3">
      <ion-icon name="square"></ion-icon>
      <ion-label>Tab 3</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab4">
      <ion-icon name="square"></ion-icon>
      <ion-label>Tab 4</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

练习: 制作tab5 标签页

- 生成页面: `ionic g page tab5`
- 配置tabs路由:
  - 移动默认生成的路由配置到tabs路由下
  - app-routing.module.ts 中的数据 移动到 tabs-routing.module.ts 中
- 配置页面
  - tabs.page.html

栈式导航路由

> 类似于vue的 route-link

生成详情页

```
ionic g page detail
```

跳转操作

```
<!--
  此页面是路由切换的, 具体查看 app.component.html 文件
  已经有ion-app了,  此处不用再写,  否则会导致 套两层 ion-app
 -->
<ion-header>
  <ion-toolbar>
    <ion-title>tab1</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <!--
    如果是vue:  <router-link to="/detail">
   -->
  <ion-button expand="block" routerLink="/detail">
    跳转到详情页
  </ion-button>
</ion-content>
```

编程方式跳转服务

```
import { Component, OnInit } from "@angular/core";
import { NavController } from "@ionic/angular";

@Component({
  selector: "app-detail",
  templateUrl: "./detail.page.html",
  styleUrls: ["./detail.page.scss"],
})
export class DetailPage implements OnInit {
  // 编程方式跳转: 需要引入跳转服务
  constructor(public router: NavController) {}

  ngOnInit() {}
}
```

返回按钮制作

```
<ion-header>
  <ion-toolbar>
    <!-- 返回操作需要导航栈: 就是个数组,其中按顺序存放了 多个组件的跳转关系 -->
    <!-- 注意!!!!  刷新操作会导致导航栈消失.  导航栈记录消失则无处返回, 就不会显示返回按钮 -
->
    <ion-buttons slot="start">
```

```
      <!-- slot: 插槽属性.  一条UI上, 通常左右各有一个空 左边是start 右侧是end  -->
      <ion-back-button></ion-back-button>
    </ion-buttons>

    <ion-title>detail</ion-title>
    <!-- 自定义返回按钮 -->
    <ion-button color="success" slot="end" (click)="router.back()">
      返回
    </ion-button>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div>{{name}}</div>
  <div>{{age}}</div>
</ion-content>
```

编程跳转

```
import { Component } from "@angular/core";
import { NavController } from "@ionic/angular";

@Component({
  selector: "app-tab1",
  templateUrl: "tab1.page.html",
  styleUrls: ["tab1.page.scss"],
})
export class Tab1Page {
  //编程式跳转 必须引入路由服务
  constructor(public router: NavController) {}

  goDetail() {
    // 跳转到详情页, 不带参数
    // this.router.navigateForward("/detail");

    // 带有参数: 使用数组  [路径, 参数对象]
    // r o 回车  . n f 回车
    this.router.navigateForward(["/detail", { name: "亮亮", age: 29 }]);
  }
}
```

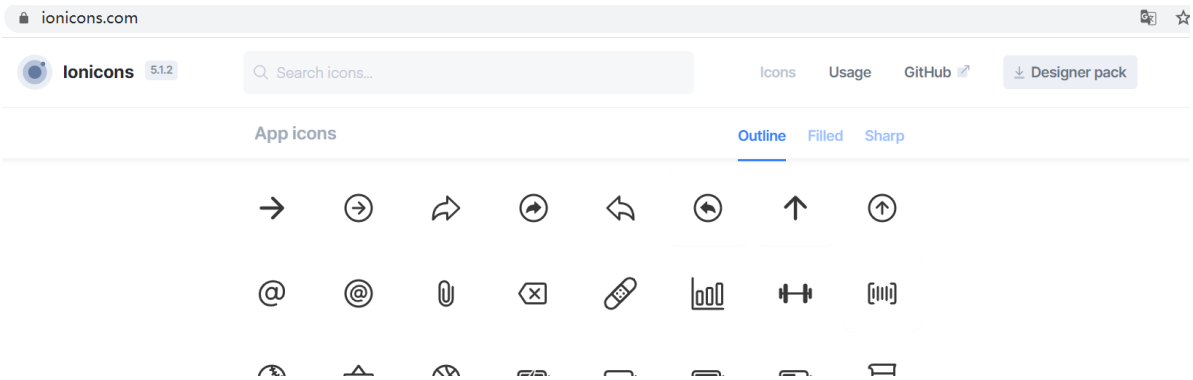
接收参数

```
import { Component, OnInit } from "@angular/core";
import { ActivatedRoute } from "@angular/router";
import { NavController } from "@ionic/angular";

@Component({
  selector: "app-detail",
  templateUrl: "./detail.page.html",
  styleUrls: ["./detail.page.scss"],
})
export class DetailPage implements OnInit {
  // 编程方式跳转: 需要引入跳转服务 NavController
```

```
  // 路由参数服务: ActivatedRoute
  constructor(public router: NavController, public route: ActivatedRoute) {}

  name = "";
  age = "";

  ngOnInit() {
    console.log(this.route);

    this.name = this.route.snapshot.params.name;
    this.age = this.route.snapshot.params.age;
  }
}
```

icon图片组件

https://ionicons.com/




```
<!-- icon 小图标 -->
<ion-header>
  <ion-toolbar>
    <ion-title>小图标</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <!-- ionic官方提供了很多常用的小图标，可以在代码中直接使用 -->
  <ion-icon name="add-circle-outline" style="font-size: 30px;"></ion-icon>

  <ion-button>
    <ion-icon name="add"></ion-icon>
  </ion-button>

  <ion-button fill="clear" color="danger">
    <ion-icon name="heart"></ion-icon>
  </ion-button>
  <br />
  <ion-icon
    name="logo-apple"
    style="font-size: 160px; color: green;"
  ></ion-icon>
```

```
</ion-content>
```

item组件


```
<ion-header>
  <ion-toolbar>
    <ion-title>item组件</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-item>
    <!-- ion-label: 块元素, 一行, 超出部分会...省略 -->
    <ion-label
      >Awesome Label Awesome Label Awesome Label Awesome Label Awesome
      Label</ion-label
    >
  </ion-item>

  <!-- 属性: button 具有点击效果 -->
  <ion-item button>
    <ion-label>可以点击</ion-label>
  </ion-item>

  <!-- 详情属性: 通过配合跳转  detail  右箭头 -->
  <ion-item button detail>
    <ion-label>可以点击</ion-label>
  </ion-item>

  <!-- 配合输入框使用: 输入框焦点状态, 下边线高亮 -->
```

```
  <ion-item>
    <ion-label>用户名:</ion-label>
    <ion-input></ion-input>
  </ion-item>

  <!-- 头像 i-item-avatar -->
  <ion-item>
    <!-- ion-avatar: 会自动把内容变圆角 -->
    <ion-avatar slot="start">
      <img
        src="https://tse2-mm.cn.bing.net/th/id/OIP.DxeVW9iSIjuEPwv6XDd6jAHaHa?
pid=Api&rs=1"
      />
    </ion-avatar>
    <ion-label>
      <h2>我是思源丫</h2>
      <p>清清，晚上咱们去吃自助餐啊?? 我请客</p>
    </ion-label>
    <ion-badge color="danger">99+</ion-badge>
  </ion-item>

  <!-- i-item-group  通讯录样式 -->
  <ion-item-group>
    <ion-item-divider color="light">
      <ion-label>A</ion-label>
    </ion-item-divider>

    <ion-item>
      <ion-label>阿宝</ion-label>
    </ion-item>
    <ion-item>
      <ion-label>阿布</ion-label>
    </ion-item>

    <ion-item-divider color="light">
      <ion-label>B</ion-label>
    </ion-item-divider>

    <ion-item>
      <ion-label>宝马</ion-label>
    </ion-item>
    <ion-item>
      <ion-label>奔驰</ion-label>
    </ion-item>
  </ion-item-group>

  <!-- 带有滑动效果 -->
  <!-- i-item-sliding -->
  <ion-list>
    <ion-item-sliding>
      <ion-item>
        <ion-label>Item</ion-label>
      </ion-item>
      <ion-item-options side="start">
        <ion-item-option (click)="favorite(item)">喜欢</ion-item-option>
        <ion-item-option color="warning" (click)="share(item)"
          >分享</ion-item-option
        >
```

```
      </ion-item-options>
      <!-- side:  start左侧,  end右侧 -->
      <ion-item-options side="end">
        <ion-item-option color="danger">删除</ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>
</ion-content>
```