前端组件

简介

本组件库的开发采用Angular8的library库功能进行实现。

Angular Library组件库说明

生成组件库

在主工程根目录下执行以下命令可自动生成组件库，自动生成ng-packagr配置文件，并下载依赖。

ng g library 【组件库名称】 --prefix=【组件库前缀】

执行之后，自动生成组件库。

目录结构为：

【主工程名称】

```
|-  projects
    |- 【组件库名称】
        |- src
            |- lib
|- src
```

自动修改主工程的配置文件angular.json，添加【组件库名称】的属性，对组件库进行相关配置，如prefix等。

自动修改主工程package.json，在devDependencies中添加ng-packagr相关依赖，并下载依赖。

自动修改主工程tsconfig.ts，在path中添加【组件库名称】的路径，也就是通过import * from '【组件库名称】'时，使用的组件库包的路径。

组件库配置文件介绍

public-api.ts

组件库最外层的把模块export的引导文件。该文件在组件build后会放在根目录下，也就是可以通过根目录import模块。

ng-package.json

ng-packagr的配置文件，自动生成，属性lib.entryFile定义了最外层引导文件的路径，默认为"src/public-api.ts"。

package.json

组件库的依赖，不能通过dependencies定义，而是通过peerDependencies定义公共组件的依赖，这样依赖包会下载到主工程的node_modules中。

tsconfig.lib.json

定义组件库的编译选项等。

araf-operetta组件库依赖管理

如果要通过npm install araf-operetta下载组件库的同时下载其它组件库依赖，需要进行如下配置：

在library的package.json中添加dependencies，**注意：**添加devDependencies并不会对其进行依赖下载。

operetta\projects\araf-operetta\package.json

```
{
  "name": "araf-operetta",
  "version": "0.0.1",
  "dependencies": {
    "devextreme": "^19.1.6",
    "devextreme-angular": "^19.1.6",
    "devextreme-intl": "^19.1.6"
  },
  "peerDependencies": {
    "@angular/common": "^8.2.7",
    "@angular/core": "^8.2.7"
  }
}
```

在library的ng-package.json中对这些依赖添加whitelistedNonPeerDependencies

operetta\projects\araf-operetta\ng-package.json

```
{
  "$schema": "../../node_modules/ng-packagr/ng-package.schema.json",
  "dest": "../../dist/araf-operetta",
  "lib": {
    "entryFile": "src/public-api.ts",
    "umdModuleIds": {
      "devextreme-angular": "devextreme-angular",
      "devextreme/localization": "devextreme/localization",
      "devextreme/ui/notify": "devextreme/ui/notify",
      "devextreme/ui/dialog": "devextreme/ui/dialog",
      "devextreme/data/data_source": "devextreme/data/data_source",
      "devextreme/ui/button": "devextreme/ui/button",
      "devextreme-angular/ui/scroll-view": "devextreme-angular/ui/scroll-view"
    }
  },
  "whitelistedNonPeerDependencies": [
    "tslib",
    "devextreme",
    "devextreme-angular",
    "devextreme-intl"
  ]
}
```

araf-operetta组件库开发规则

公共组件Component

- 公共组件路径为：/araf-operetta/src/lib/components/【组件类型】/【公共组件】，其中【组件类型】根据组件本身类型进行选择。
  目前组件类型有：layout（布局），input（输入），button（按钮），toolbar（工具栏）。如TicketNo组件属于输入类型，那么就放在input下。
- 在组件库中添加模块/组件，可以使用命令ng g m components/【组件类型】/【组件名称】 --project=araf-operetta创建组件模块，再使用ng g c components/【组件类型】/【组件名称】 --module=【组件名称（模块名）】 --project=araf-operetta创建组件。
- 每个公共组件都需要有单独的【组件名】.module.ts文件，且export的**module名必须以Araf开头**，如：ArafTicketNoModule。
- 组件类中所有import的路径，都需要使用相对路径，如：import { ArafCommonService } from '../../../utils/araf-common.service';
- 组件Module需要在/araf-operetta/src/lib/components/**【组件类型】/index.ts**中进行export，如：export * from './ticket-no/ticket-no.module';

公共管道Pipe

- 公共服务的路径为：/araf-operetta/src/lib/pipes/
- 在组件库中添加管道，可以使用命令ng g p pipes/【管道名】 --module=araf-pipe --project=araf-operetta
- 公共管道**@Pipe的name必须以“araf_”开头**，如：araf_translate。
- 类中所有import的路径，都需要使用相对路径，如：import { ArafCommonService } from '../utils/araf-common.service';
- 公共管道类需要添加在araf-pipe.module.ts中@NgModule的exports中，如：
```
exports: [
    TranslatePipe
]
```

公共服务Service

- 公共服务的路径为：/araf-operetta/src/lib/utils/
- 在组件库中添加服务，可以使用命令ng g s utils/【公共服务名】 --project=araf-operetta
- 公共服务**文件名必须以“araf-”开头**，如：araf-common.service.ts
- 类中所有import的路径，都需要使用相对路径，如：import { ArafCommonService } from './araf-common.service';
- 公共服务需要要在/araf-operetta/src/lib/utils/index.ts中进行export，如：export * from './araf-common.service';

araf-operetta组件库打包

打包

在主工程中使用命令

ng build araf-operetta

包路径为operetta/dist/araf-operetta

跨版本打包

araf-operetta组件库使用的是Angular8，如果要应用于Angular7的工程，在通过ng build araf-operetta打包之后，需要修改包内文件：**araf-operetta/fesm5/araf-operetta.js**，将所有inject替换为inject、所有defineInjectable替换为defineInjectable，即**去掉所有**。

否则在Angular7项目中使用，启动时会报错"export 'inject' was not found in '@angular/core'。打开页面，控制台会报错Uncaught TypeError: Object(...) is not a function

araf-operetta组件库使用

安装

npm install araf-operetta

使用组件Component/管道Pipe

在module中对使用的组件模块进行import

example.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { CommonControlsComponent } from '.
/common-controls/common-controls.component';
import {
  ArafSingleCardModule,
  ArafTicketNoModule,
  ArafPipeModule
} from 'araf-operetta';
import { NotifyComponent } from './notify
/notify.component';

@NgModule({
  declarations: [CommonControlsComponent],
  imports: [
    CommonModule,
    ArafSingleCardModule,
    ArafTicketNoModule,
    ArafPipeModule
  ],
  exports: []
})
export class ExampleModule { }
```

使用服务Service

在ts中注入公共服务

ts

```
import { ArafCommonService } from 'araf-
operetta';

constructor(private commonService:
ArafCommonService) { }

ngOnInit() {
   if (this.commonService.isEmpty(this.value))
{
      console.log("empty");
   }
}
```