راهنمای زیر ساخت نرم افزار در بخش فرانت اند

1401/11/16

مقدمه:

بعد از راه اندازی سورس کد مربوط به بخش فرانت اند و نصب پکیج های مورد نیاز میتوان با توجه به ساختار برنامه کامپوننت های موجود را تولید کنید.

البته این ساختار در آینده ممکن هست دچار تغییرات و بهبود هایی شود که میتوان این تغییرات را در نسخه های بعدی این راهنما در نظر گرفت.

در ادامه مطلب به بررسی سرفصل های زیر میپردازیم :

- آشنایی اولیه با ساختار فولدر ها
- انواع کامپوننت های قابل پیاده سازی در برنامه
- نحوه تنظیم مقادیر اولیه در environment
- بررسی ماژول admin
- بررسی ماژول shared
- بررسی و آشنایی با ماژول finance
- نحوه ایجاد یک کامپوننت GridExpolorer
- نحوه ایجاد کامپوننت AutoGeneratedGrid
- گزارش فوری
- نکات مهم

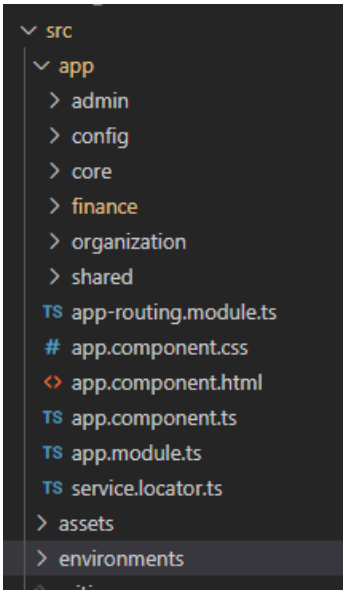
آشنایی اولیه با ساختار فولدر ها :

پروژه در بخش فرانت اند از فولدر هایی که در تصویر زیر آمده است تشکیل شده است هر کدام از این فولدر ها به منظور خاصی وبرای نگهداری کامپوننت های مربوطه ایجاد شده است.

برای ایجاد هر کامپوننت فولدر هایی در نظر گرفته شده است که در ادامه به بررسی جزییات هر کدام می پردازیم.

نکته مهمی که باید در طول تولید فرم ها رعایت شود چند زبانه بودن سیستم میباشد به طور کلی همه ریسورس ها در فولدر assets نگهداری میشود این ریسورس ها شامل تصویر ، فایل های css ، js و json و ... می باشد .

فایل های ریسورس مربوط به چند زبانه بودن در آدرس assets/i18n/ نگهداری می شود.



ماژول admin کامپوننت ها و موارد مربوط به منو های کاربران و سازمان ، ماژول config مربوط به کامپوننت های تنظیمات ، ماژول core حاوی تعدادی مدل مدل و سرویس برای عملیات های پایه ای مربوط به کاربران و عملیات های لاگین ، ماژول organization مربوط به منوی سازمان در برنامه می باشد.

مهمترین ماژول ها در حال حاضر ماژول های finance و shared می باشد که به ترتیب برای کامپوننت های حسابداری و کامپوننت های سفارشی شده برای برنامه در نظر گرفته شده است.

انواع کامپوننت های قابل پیاده سازی در برنامه

در حال حاضر دو نوع کامپوننت در برنامه قابل پیاده سازی هستند کاپوننت هایی که دارای یک کنترل TreeView در کنار گرید می باشند که در اینجا GridExplorer شناخته میشود و مدل دوم کامپوننت هایی که شامل یک گرید در فرم اصلی می باشد و فرم های مودال بر روی گرید مربوطه باز میشود.

به ترتیب کامپوننت های AutoGridExplorerComponent و AutoGeneratedGridComponent مربوط به این کامپوننت های می باشد.

نحوه تنظیم مقادیر اولیه در environment

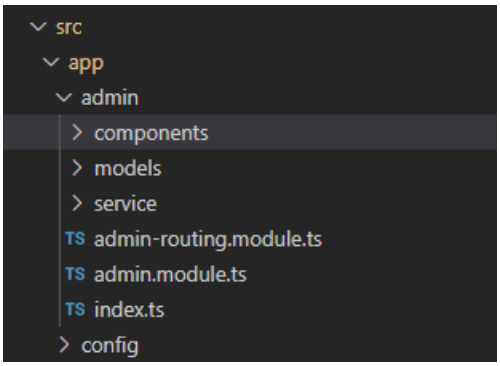
```
export const environment = {
  production: false,
  BaseUrl: "http://localhost:8801",
  LicenseServerUrl: "http://localhost:7473",
  InstanceKey:
    "bIiGXi1RadcLQ+H7EsAX3Sa+uGHQER1xrsXN7wa9Udb3Xp3YeZ0zwsGS/nOYCjbzG4btY1pDYWWVGi7taArIQ7Db
G0JKpTglUvMlYySMKkbmH7EsFS9mjpGHk6QME3Vor6FWcOMc5e4Ka/OMu23wkvlZ583ICrtj5oQZQXX5oFpMME9r+9z2p
ctAw378koGWZLc7A8etqUC6dozMOxAxww==",
  Version: "1.2.1373",
  SessionAliveInterval: 300000,
```

```
};
```

در فایل های مربوط به environment تنظیمات مربوط به اتصال به سرویس اصلی ، سرویس لایسنس و همچنین مقدار key مربوط به لایسنس ارایه شده را میتوان وارد کرد. در صورت ورود مقادیر صحیح و در دسترس بودن سرویس ها شما میتوانید به برنامه لاگین کنید.

بررسی ماژول admin

در واقع هر فولدر شامل مواردی هست که در تصویر مشخص می باشد :



فولدر components محل نگهداری کامپوننت های مربوط به زیر سیستم مربوطه میباشد فولدر models محل نگهداری مدل های تعریف شده برای بخش client و فولدر service مربوط به سرویس های نوشته شده برای برقراری ارتباط با سرویس های api می باشد.

فایل admin.module.ts ماژول اصلی مربوط به زیر سیستم admin می باشد که کامپوننت های این زیر سیستم در این فایل declare شده اند.

بنابراین برای ایجاد یک کامپوننت جدید تمامی این موارد باید ایجاد شود.

بررسی ماژول shared

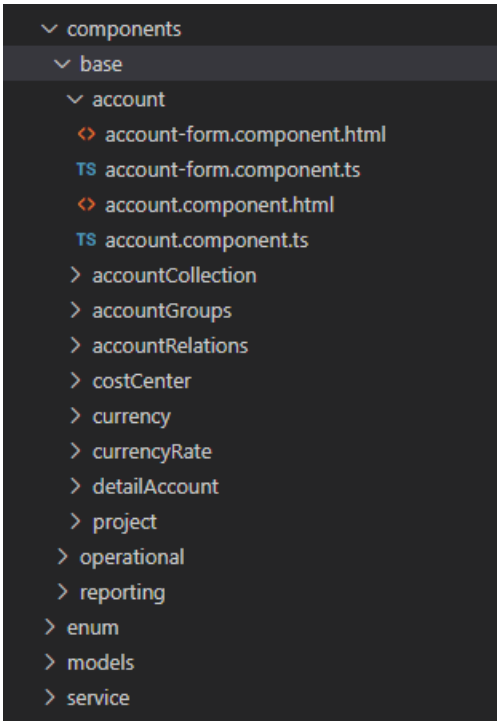
این ماژول برای نگهداری اجزاری مشترک بین ماژول های برنامه میباشد به عنوان مثال کامپوننت های مشترک در بخش finance یا حتی کنترل هایی مرتبط با UI یا کلاس های پایه در این ماژول نگهداری میشود در این ماژول نگهداری میشود.

بررسی و آشنایی با ماژول finance

در این ماژول کامپوننت های مربوط به ساب سیستم حسابداری شامل فرم های اطلاعات پایه ، گزارش ها و ... نگهداری میشود.

نحوه ایجاد یک کامپوننت GridExpolorer

برای ایجاد کاپوننت جدید شما نیاز به ایجاد سرویس ، مدل ، کلاس API و خود کامپوننت ها دارید به عنوان مثال میتوان این نامگذاری را برای کامپوننت ها در نظر گرفت :



همانطور که در شکل مشخص میباشد مدل و سرویس مربوطه در فولدر های مشخص شده ساخته میشود.

همچنین کلاس api که شامل آدرس های api مربوط به سرویس میباشد را در فولدر service/api ایجاد کنید.

کامپوننت دلخواه خود را در فولدر معین شده با درستور ng g c تولید کنید پس از آن مطابق کد های زیر را در آن تغییر دهید.

حتما از کلاس AutoGridExplorerComponent ارث بری انجام شود

```
export class AccountComponent
  extends AutoGridExplorerComponent<Account>
  implements OnInit
{
```

مطابق شکل زیر عمل کنید خاصیت های entityName و viewId را مقدار دهی کنید همچنین خاصیت getDataUrl آدرس api مربوط به سرویس را در خود نگهداری میکند.

در آخر با دستور reloadGrid دیتا های مورد نیاز در خاصیت ها پر میشود و datagrid اصلی پر میشود.

```
ngOnInit(): void {
    this.entityName = Entities.Account;
    this.viewId = ViewName[this.entityTypeName];

    this.getDataUrl = AccountApi.EnvironmentAccounts;
    this.treeConfig = this.getViewTreeSettings(this.viewId);
    this.getTreeNode();
    this.reloadGrid();
```

برای باز کردن dialog های مودال مطابق دستور های زیر عمل کنید:

```
 this.dialogRef = this.dialogService.open({
          title: editorTitle,
          content: AccountFormComponent,
        });

        this.dialogModel = this.dialogRef.content.instance;
        this.dialogModel.parent = this.parent;
        this.dialogModel.model = this.editDataItem;
        this.dialogModel.isNew = isNew;
        this.dialogModel.errorMessages = undefined;

        this.dialogRef.content.instance.save.subscribe((res) => {
          this.saveHandler(res, isNew);
        });

        const closeForm = this.dialogRef.content.instance.cancel.subscribe(
          (res) => {
            this.dialogRef.close();
          }
        );
```

برای فرم های جزییات حتما از کلاس DetailComponent ارث بری انجام شود.

```
export class AccountFormComponent extends DetailComponent implements OnInit {
```

نکته مهم : با توجه به اینکه متادیتا های هر گامپوننت با توجه viewId تهیه و در دیتابیس نگهداری میشود شما نیازی به تعریف FormGroup و FormControl به صورت دستی ندارید فقط کافی هست متغیر editForm را به یک تگ form متصل کنید.

برای جزییات بیشتر میتوانید کد های موجود را بررسی نمایید.

نحوه ایجاد یک کامپوننت AutoGeneratedGrid

به طور کلی کامپوننت هایی که نمیخواهیم در سمت راست آن TreeView ایجاد شود و حالت درختی ندارد میتوان از این کلاس ارث بری کرد به عنوان مثال گزارشات و یا فرم نمایش رویداد ها و یا فرم اسناد مالی .

این کامپوننت های شامل یک گرید اصلی میباشد که با توجه به viewId و متادیتا ها پر میشود یعنی ستون های گرید با توجه به متادیتا ها فراهم می شود . این متادیتا بعد از لاگین برا کامپوننت ها فراهم میشود.

مطابق شکل زیر میتوان یک کامپوننت AutoGeneratedGrid ایجاد کرد :

```
export class VoucherComponent
  extends AutoGeneratedGridComponent
  implements OnInit
{
```

گزارش فوری :

برای ایجاد گزارش فوری باید در جدول های Report و LocalReport رکورد مورد نظر را ایجاد کرده این کار را میتوان با ابزار تهیه شده در که در سورس موجود هست انجام داد.

سپس کد های زیر در فایل ts اضافه میکنیم

```
 @ViewChild(GridComponent, {static: true}) grid: GridComponent;
  @ViewChild(ViewIdentifierComponent, {static: true}) viewIdentity: ViewIdentifierComponent;
  @ViewChild(ReportViewerComponent, {static: true}) viewer: ReportViewerComponent;
  @ViewChild(ReportManagementComponent, {static: true}) reportManager:
ReportManagementComponent;
  @ViewChild(QuickReportSettingComponent, {static: true}) reportSetting:
QuickReportSettingComponent;
```

در فایل html کامپوننت های زیر را استفاده می کنیم :

```
<view-identifier [ViewID]="viewId">
  <report-param ParamName="fromDate" [ParamValue]="fromDate"></report-param>
  <report-param ParamName="toDate" [ParamValue]="toDate"></report-param>
  <report-param ParamName="fromNo" [ParamValue]="fromVoucher"></report-param>
  <report-param ParamName="toNo" [ParamValue]="toVoucher"></report-param>
</view-identifier>
<report-management
  [ViewIdentity]="viewIdentity"
  [Grid]="grid"
  [Sort]="sort"
  [Filter]="currentFilter"
  [QuickFilter]="reportQuickFilter"
  [RowData]="rowData"
  [DefaultServiceUrl]="getDataUrl"
></report-management>
<report-setting></report-setting>
```

با استفاده از توابع زیر میتوانید گزارش چاپی یا فرم تنظیمات گزارش چاپی را در برنامه نمایش دهیم :

```
public showReport() {

    if (this.validateReport()) {
      if (!this.reportManager.directShowReport()) {
        this.showMessage(this.getText("Report.PleaseSetQReportSetting"));
        this.showReportSetting();
      }
    }
  }

  public validateReport() {
    if (!this.rowData || this.rowData.total == 0) {
      this.showMessage(this.getText("Report.QuickReportValidate"));
      return false;
    }
    return true;
  }

  public showReportSetting() {
    if (this.validateReport()) {
      this.reportSetting.showReportSetting(this.gridColumnsRow, this.entityTypeName,
this.viewId, this.reportManager);
    }
  }
```

متغیر های **DefaultFilter** و **QuickFilter** در گزارشات

متغیر DefaultFilter مقدار شروط مربوط به ستون های گرید را در خود نگهداری میکند و معمولا به صورت خودکار مقداردهی و محاسبه میشود ولی متغیر quickFilter فیلتر های سریع می باشد مثلا در گزارشات فیلتر هایی که پایین هر جدول وجود دارد فیلتر های سریع هست و باید در این متغیر اضافه شود.

نکات مهم :

- حتما از وجود متادیتا ها در دیتابیس سیستمی اطمینان حاصل کنید
- برای گزارش فوری حتما باید رکورد های گزارش در جداول Report و LocalReport درج شوند.
- حتما در enum به نام ViewName اسم view و کد آن را وارد کنید این کد باید با کد view در دیتابیس برابر باشد.
- اگر گزارش فوری شامل پارامتر هایی مثل از تاریخ تا تاریخ میباشد میتوانید از کد زیر برای ارسال پارامتر ها استفاده کنید. البته این پارامترها در جدول Parameter در داخل DB تعریف شده اند.

```
<view-identifier [ViewID]="viewId">
  <report-param ParamName="fromDate" [ParamValue]="fromDate"></report-param>
  <report-param ParamName="toDate" [ParamValue]="toDate"></report-param>
  <report-param ParamName="fromNo" [ParamValue]="fromVoucher"></report-param>
  <report-param ParamName="toNo" [ParamValue]="toVoucher"></report-param>
</view-identifier>
```

- سرویس پایه برای کار با localStorage و sessionStorage به نام BrowserStorageService وجود دارد که در همه جا قابل استفاده میباشد.
- برای اینکه مقادیر فیلتر های سریع تا زمانی که کاربر در محیط برنامه هست حفظ کنید از دکوراتور @Persist استفاده کنید.