User Controls in Angular

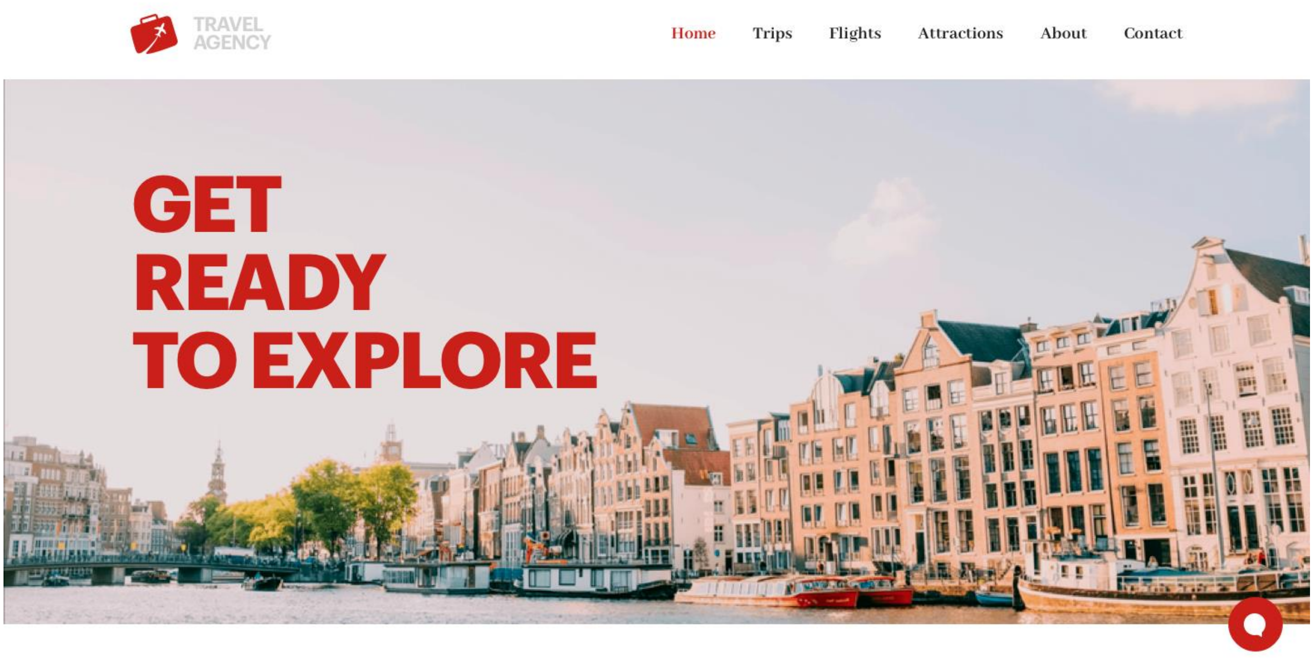
GeneXus™

Previous videos showed several screen controls that help in building the user interface, and the way to enhance the app's design with definitions in a Design System Object, as well as how to import a design already made by a designer in Sketch.

This video shows that, in addition to the predefined screen controls available in the toolbar, it is also possible to create your own controls to enrich the user experience even further.

Importing resources from UI providers

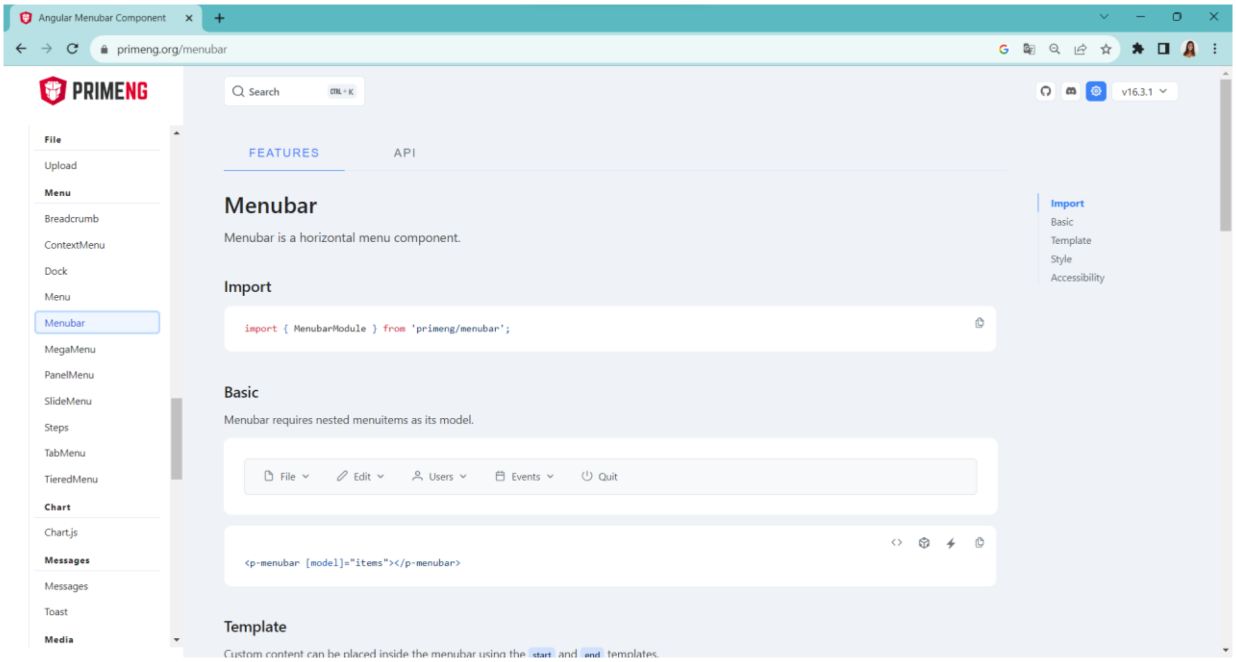
GeneXus allows you to create user controls from controls built by designers, or controls available on platforms of providers of User Interface resources. These may be either native components of the Angular framework –such as PrimeNG, AngularMaterial or Material-UI-, or HTML, CSS and JavaScript resources from generic providers like SemanticUI, VanillaFramework, or Bootstrap component libraries.



In the design received from the designer, there is an option menu in the upper right.

You will now build the menu with a user control provided by PrimeNG.

Defining the User Control to be created



Menubar

Menubar is a horizontal menu component.

Import

```
import { MenubarModule } from 'primeng/menubar';
```

Basic

Menubar requires nested menuitems as its model.

```
<p-menubar [model]="items"></p-menubar>
```

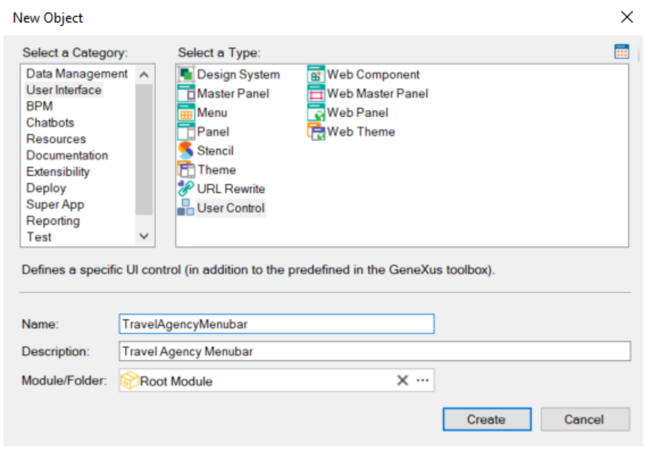
Template

Custom content can be placed inside the menubar using the start and end templates.

In the provider page you will find several menu examples, where the most suitable in your case is the Menubar. Here you can see the page with the data relative to that control.
Below the Features tab we find, for example, general information about the control, as well as how the menu would look like and the HTML needed to implement it.

The procedure for creating the user control in GeneXus that will be included in panel objects is similar to that of a web app built with web panels, as it was shown in other videos. You will obtain the control’s HTML and add it in the user control to be created. Then you import other resources such as CSS libraries and so on.

Creating the User Control object



New Object

Select a Category: Data Management, User Interface, BPM, Chatbots, Resources, Documentation, Extensibility, Deploy, Super App, Reporting, Test

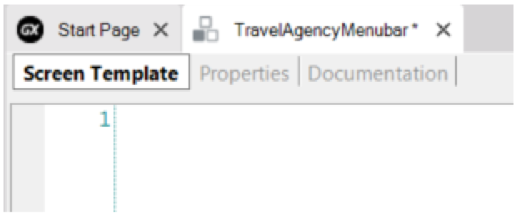
Select a Type: Design System, Master Panel, Menu, Panel, Stencil, Theme, URL Rewrite, User Control, Web Component, Web Master Panel, Web Panel, Web Theme

Defines a specific UI control (in addition to the predefined in the GeneXus toolbox).

Name: TravelAgencyMenubar

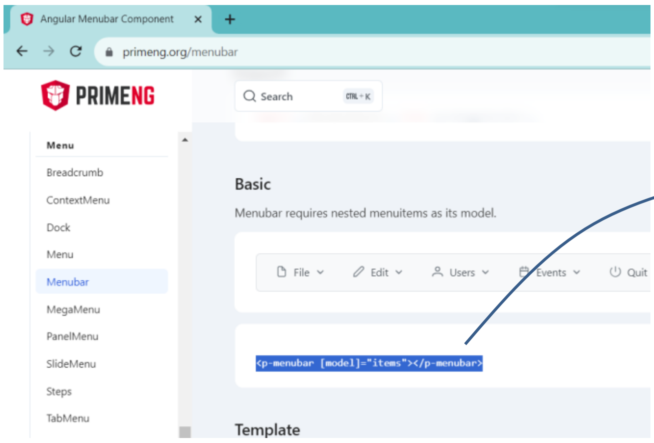
Description: Travel Agency Menubar

Module/Folder: Root Module



Create an object of the User Control type and call it TravelAgencyMenubar. The object has 2 sections with which you will be working, namely: Screen Template, where you will define the control’s html code, and Properties, where you will assign values to some of the elements in the HTML.

Copy and paste HTML code on the control

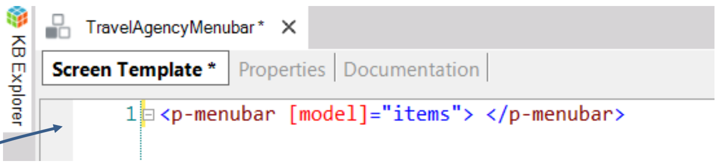


Basic

Menubar requires nested menuitems as its model.

```
<p-menubar [model]="items"></p-menubar>
```

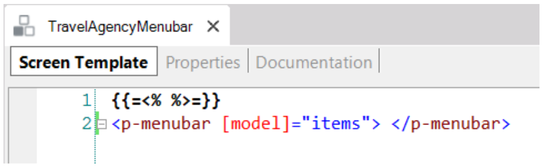
Template



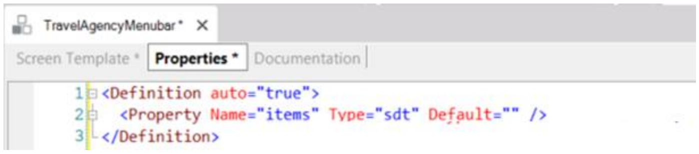
```
<p-menubar [model]="items"> </p-menubar>
```

If we go back to the PrimeNG page and go to Basic, we can see the HTML of the Menubar control, so we copy and paste it into the Screen Template section of our control.

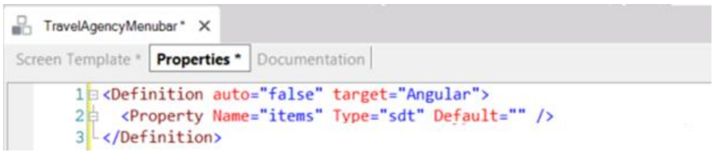
Substitute fixed data with variable elements



```
{{=<% %>=}}
<p-menubar [model]="items"> </p-menubar>
```



```
<Definition auto="true">
  <Property Name="items" Type="sdt" Default="" />
</Definition>
```



```
<Definition auto="false" target="Angular">
  <Property Name="items" Type="sdt" Default="" />
</Definition>
```

Now, substitute the fixed data included in the HTML with elements that will allow you to load them dynamically with your own data, either fixed or from the database.

Before the HTML that we pasted in the Screen Template, let's add the following: bracket, bracket, equal to, lower than, percentage; percentage, greater than, equal to, bracket, bracket.

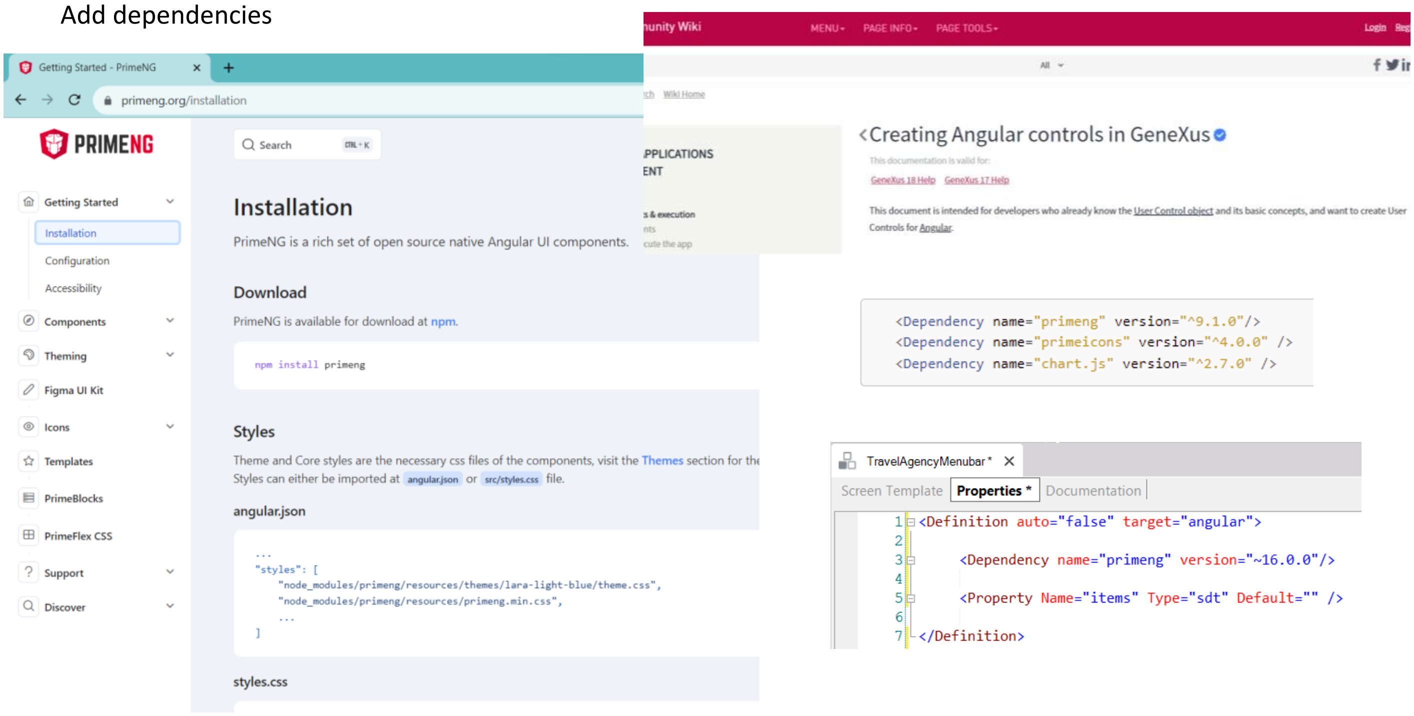
In this way, we indicate the generator to interpret all the code written there as Angular code.

We save and go to the Properties tab. We complete the code in this section by adding the property corresponding to the variable we added in the Screen Template and setting its data type as "sdt" because we are going to load it with the SDT from the menu.

The data types that you may use are shown in the general documentation of the Wiki User Controls.

Now change the Definition clause with auto="false" and add target = "Angular". This is required for the UC to remain visible in order to be added in a panel object.

Add dependencies



Installation

PrimeNG is a rich set of open source native Angular UI components.

Download

PrimeNG is available for download at npm.

```
npm install primeng
```

Styles

Theme and Core styles are the necessary css files of the components, visit the Themes section for the
Styles can either be imported at angular.json or src/styles.css file.

angular.json

```
...
"styles": [
    "node_modules/primeng/resources/themes/lara-light-blue/theme.css",
    "node_modules/primeng/resources/primeng.min.css",
    ...
]
```

styles.css

Creating Angular controls in GeneXus

This documentation is valid for:

This document is intended for developers who already know the User Control object and its basic concepts, and want to create User Controls for Angular.

```
<Dependency name="primeng" version="^9.1.0"/>
<Dependency name="primeicons" version="^4.0.0" />
<Dependency name="chart.js" version="^2.7.0" />
```

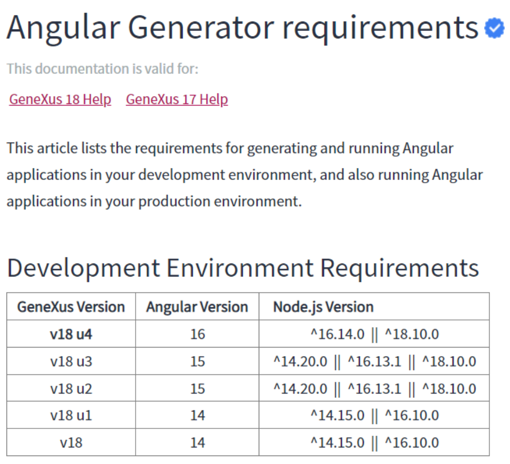
```
<Definition auto="false" target="angular">

    <Dependency name="primeng" version="~16.0.0"/>

    <Property Name="items" Type="sdt" Default="" />

</Definition>
```

Now you must add several lines to import components from the provider that will be needed to properly interpret the control.

The first thing to add are the dependencies of the packages that will be imported. If we go to PrimeNG’s website: primeng.org and browse below Getting Started, a page opens with a user's guide for the library controls. Even though this guide depends on each vendor, they all have documentation showing how to get their components.

In the Installation section, à Download indicates which package should be installed with npm. This package is the one we need to declare in our user control as a dependency.

If we go to the wiki page that describes how to use a user control in Angular and go to the Dependencies section, we see the syntax we should use. We add the dependency to the Properties section of our User Control with the data from the vendor’s page.



Angular Generator requirements

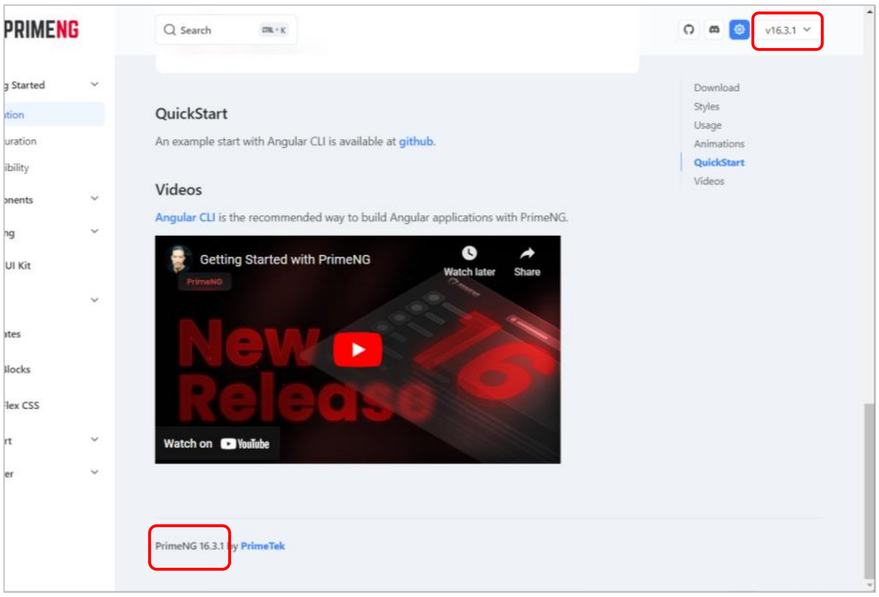
This documentation is valid for:

GeneXus 18 Help GeneXus 17 Help

This article lists the requirements for generating and running Angular applications in your development environment, and also running Angular applications in your production environment.

Development Environment Requirements

GeneXus Version	Angular Version	Node.js Version
v18 u4	16	^16.14.0 || ^18.10.0
v18 u3	15	^14.20.0 || ^16.13.1 || ^18.10.0
v18 u2	15	^14.20.0 || ^16.13.1 || ^18.10.0
v18 u1	14	^14.15.0 || ^16.10.0
v18	14	^14.15.0 || ^16.10.0



v16.3.1

QuickStart

An example start with Angular CLI is available at github.

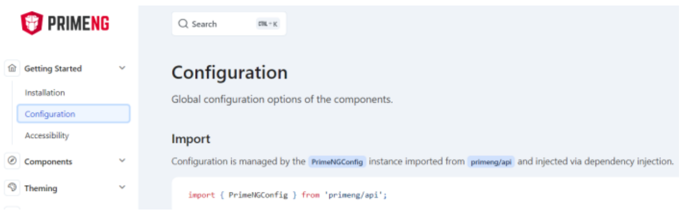
Videos

Angular CLI is the recommended way to build Angular applications with PrimeNG.

PrimeNG 16.3.1 by PrimeTek

Note that we must take into account the version of Angular we are using because the controls we reference must be available for that version. In the wiki page that shows the requirements of the Angular generator, we can find the version used by GeneXus. In addition, the version used by the vendor for the controls is shown in the upper right corner and also in the footer of the control page.

Add resource imports



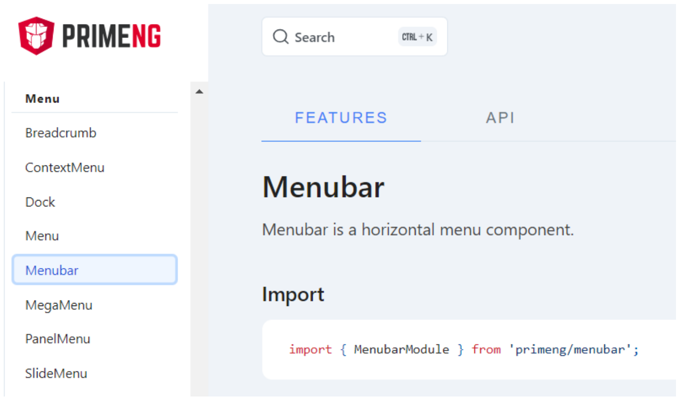
Configuration

Global configuration options of the components.

Import

Configuration is managed by the PrimeNGConfig instance imported from primeng/api and injected via dependency injection.

```
import { PrimeNGConfig } from 'primeng/api';
```

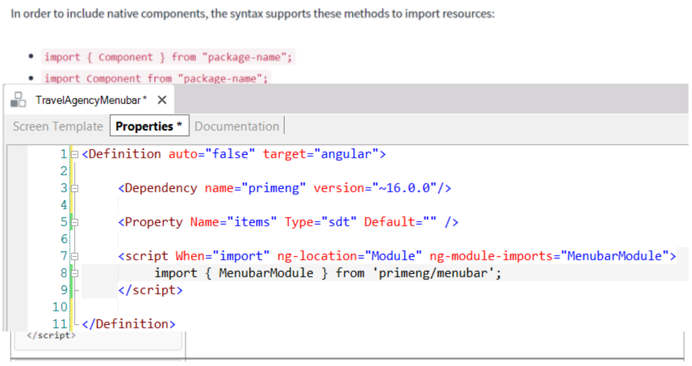


Menubar

Menubar is a horizontal menu component.

Import

```
import { MenubarModule } from 'primeng/menubar';
```



In order to include native components, the syntax supports these methods to import resources:

- `import { Component } from "package-name";`
- `import Component from "package-name";`

```
<Definition auto="false" target="angular">

    <Dependency name="primeng" version="~16.0.0"/>

    <Property Name="items" Type="sdt" Default="" />

    <script When="import" ng-location="Module" ng-module-imports="MenubarModule">
        import { MenubarModule } from 'primeng/menubar';
    </script>

</Definition>
```

If we go to the "Configuration" section of Getting Started, in the Import part we see the module we need to import. Here is a generic example, so to know which modules we need for the Menubar control, we go to the information page of the Menubar control, which is further down in the left menu, below the Menu section.

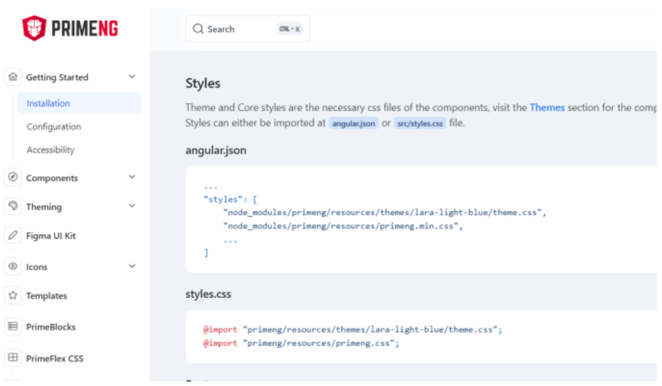
Note that the module we should import is the MenubarModule.

Front-end generators, such as Angular, contribute with advanced ways of including external resources using the “import” sentence. This declaration may be used both for including JavaScript modules and for including other types of resources, like images, SVG files or stylesheets, through a package as webpack.

In the wiki you will find the syntax to use in GeneXus for the import. First there is a description of the various syntaxes of the import command and below are some examples of their use. You may choose any because in this case there are no special requirements as to where the code generated must be.

Adapt the wiki example to import the MenubarModule module from primeng/menubar and add it to the Properties window.

Add styles



Styles

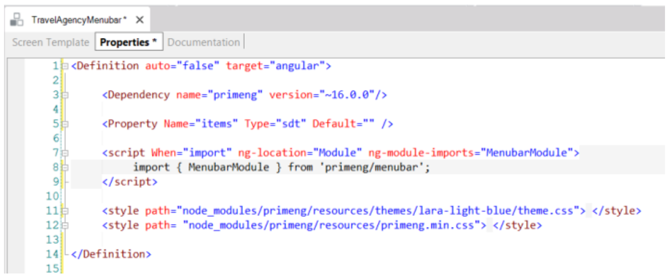
Theme and Core styles are the necessary css files of the components, visit the Themes section for the comp
Styles can either be imported at angular.json or src/styles.css file.

angular.json

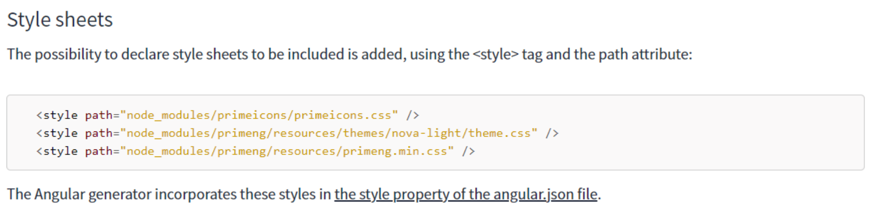
```
...
"styles": [
    "node_modules/primeng/resources/themes/lara-light-blue/theme.css",
    "node_modules/primeng/resources/primeng.min.css",
    ...
]
```

styles.css

```
@import "primeng/resources/themes/lara-light-blue/theme.css";
@import "primeng/resources/primeng.css";
```



```
<Definition auto="false" target="angular">

    <Dependency name="primeng" version="~16.0.0"/>

    <Property Name="items" Type="sdt" Default="" />

    <script When="import" ng-location="Module" ng-module-imports="MenubarModule">
        import { MenubarModule } from 'primeng/menubar';
    </script>

    <style path="node_modules/primeng/resources/themes/lara-light-blue/theme.css"> </style>
    <style path= "node_modules/primeng/resources/primeng.min.css"> </style>

</Definition>
```

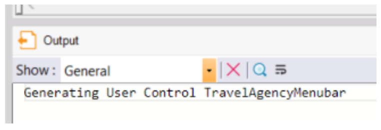


Style sheets

The possibility to declare style sheets to be included is added, using the <style> tag and the path attribute:

```
<style path="node_modules/primeicons/primeicons.css" />
<style path="node_modules/primeng/resources/themes/nova-light/theme.css" />
<style path="node_modules/primeng/resources/primeng.min.css" />
```

The Angular generator incorporates these styles in the style property of the angular.json file.



```
Generating User Control TravelAgencyMenubar
```

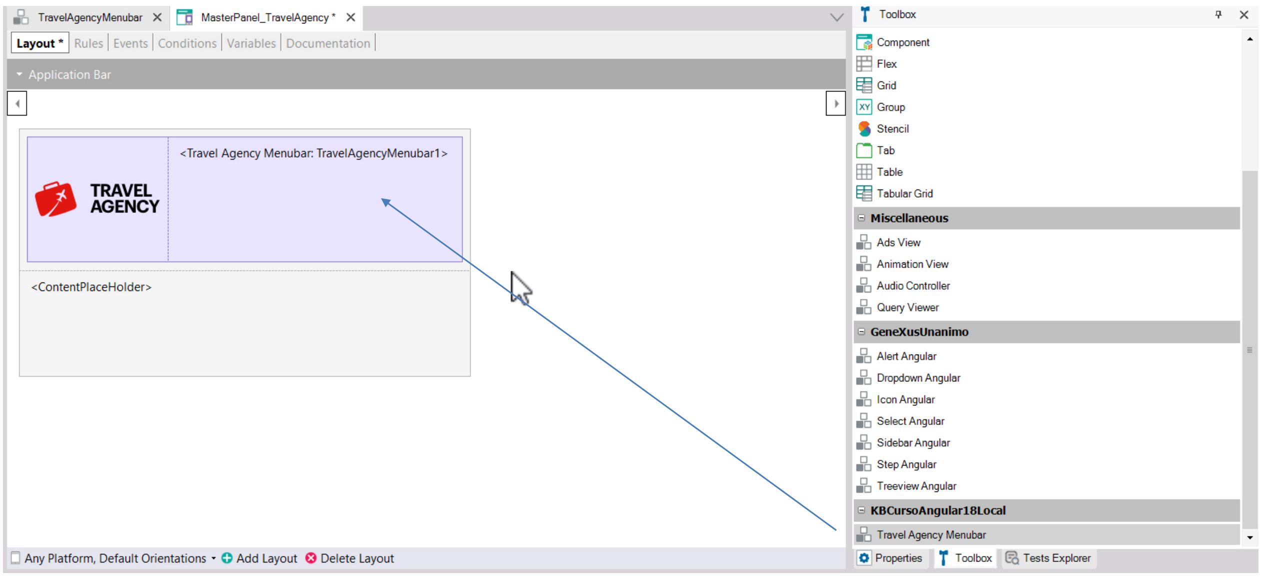
We go back again to the "Installation" page of Getting Started and see that it says we should also load CSS code.
Go to the wiki again to see how to write that in GeneXus using the style clause and detailing the path where the CSS necessary will be located.

Now that the definition of the User Control is complete, do Save and you will see that the TravelAgencyMenubar user control has been generated.

Now proceed to include it in a panel object.

GeneXus

Insert the User Control created in the MasterPanel

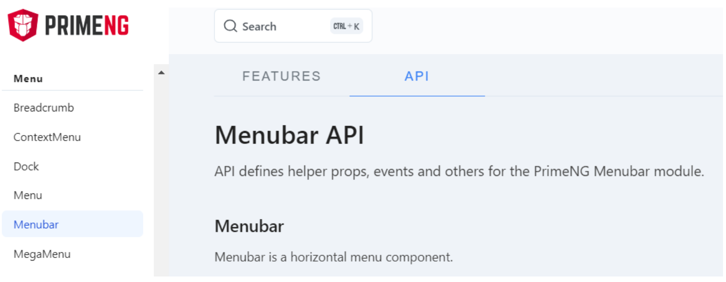


The best place to include a menu is in a Master Panel object so that the menu is present in all of the app’s screens, ensuring quick access to the various parts. So, open the MasterPanel_TravelAgency object you had built.

You will see that the TravelAgencyMenubar user control you created appears on the toolbar, so now you drag it to the form.

A user control called TravelAgencyMenubar1 is created, and since you don’t need the table to the right of the logo, you just simplify the design a bit.

Create the SDT with the structure required

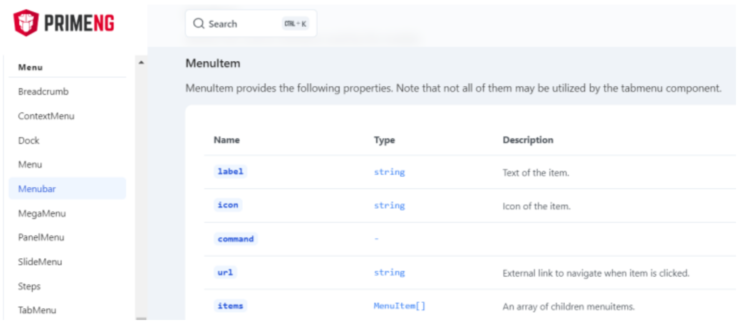


Menubar API

API defines helper props, events and others for the PrimeNG Menubar module.

Menubar

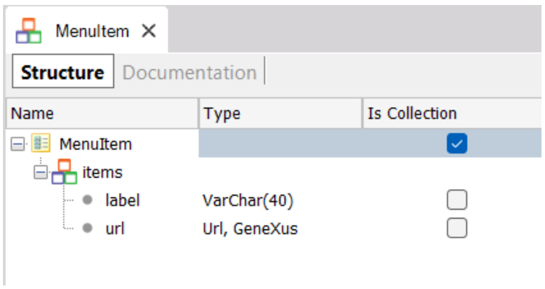
Menubar is a horizontal menu component.



MenuItem

MenuItem provides the following properties. Note that not all of them may be utilized by the tabmenu component.

Name	Type	Description
label	string	Text of the item.
icon	string	Icon of the item.
command	-	
url	string	External link to navigate when item is clicked.
items	MenuItem[]	An array of children menuitems.



Name	Type	Is Collection
MenuItem		✓
items		
label	VarChar(40)	☐
url	Url, GeneXus	☐

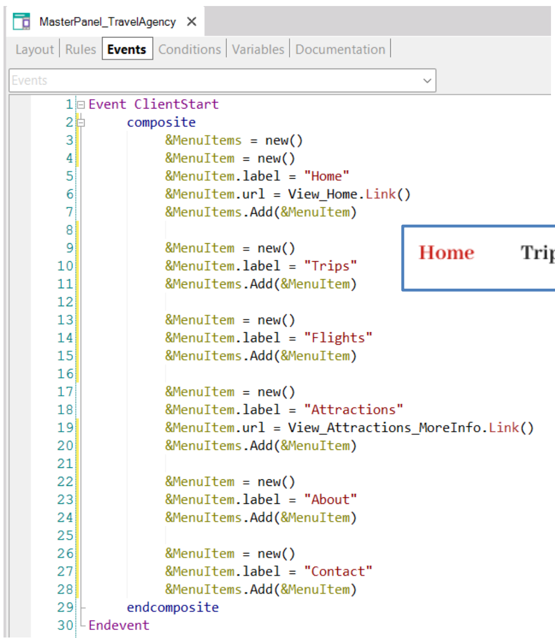
As mentioned, the menu items are loaded using an SDT.

To know the structure we have to give to the SDT, we return to the Menubar control documentation and in the API tab we see the MenuItem documentation with its properties. There we see that a menu item has an ID, a label, an icon, a command, a URL, etc.

So we can create an SDT called MenuItem, of collection type. In each item it has a label, of Character type, and a URL, which will contain the URL of the object that will be invoked when we select the menu.

Every API of each provider is different, so you must refer to the provider documentation to see how you set up your User Control and which structures you must use to load the data needed.

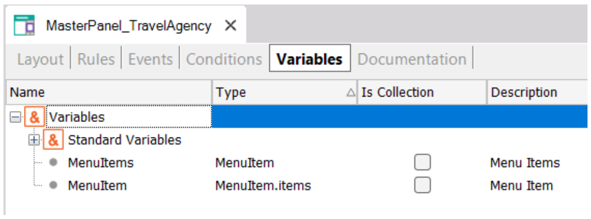
Load the SDT in the MasterPanel events


MasterPanel_TravelAgency

Layout | Rules | **Events** | Conditions | Variables | Documentation

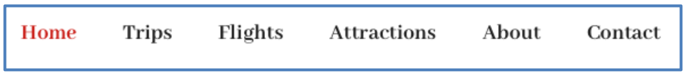
Events

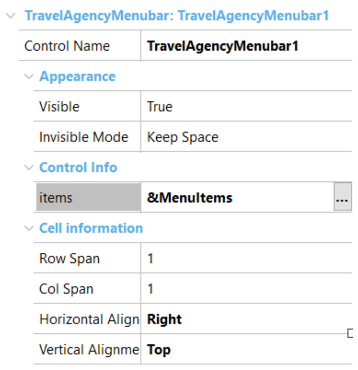
```
Event ClientStart
    composite
        &MenuItems = new()
        &MenuItem = new()
        &MenuItem.label = "Home"
        &MenuItem.url = View_Home.Link()
        &MenuItems.Add(&MenuItem)

        &MenuItem = new()
        &MenuItem.label = "Trips"
        &MenuItems.Add(&MenuItem)

        &MenuItem = new()
        &MenuItem.label = "Flights"
        &MenuItems.Add(&MenuItem)

        &MenuItem = new()
        &MenuItem.label = "Attractions"
        &MenuItem.url = View_Attractions_MoreInfo.Link()
        &MenuItems.Add(&MenuItem)

        &MenuItem = new()
        &MenuItem.label = "About"
        &MenuItems.Add(&MenuItem)

        &MenuItem = new()
        &MenuItem.label = "Contact"
        &MenuItems.Add(&MenuItem)
    endcomposite
Endevent
```


MasterPanel_TravelAgency

Layout | Rules | Events | Conditions | **Variables** | Documentation

Name	Type	Is Collection	Description
Variables			
Standard Variables			
MenuItems	MenuItem	☐	Menu Items
MenuItem	MenuItem.items	☐	Menu Item




TravelAgencyMenubar: TravelAgencyMenubar1

Control Name	**TravelAgencyMenubar1**
Appearance	
Visible	True
Invisible Mode	Keep Space
Control Info	
items	**&MenuItems**
Cell information	
Row Span	1
Col Span	1
Horizontal Align	**Right**
Vertical Alignme	**Top**

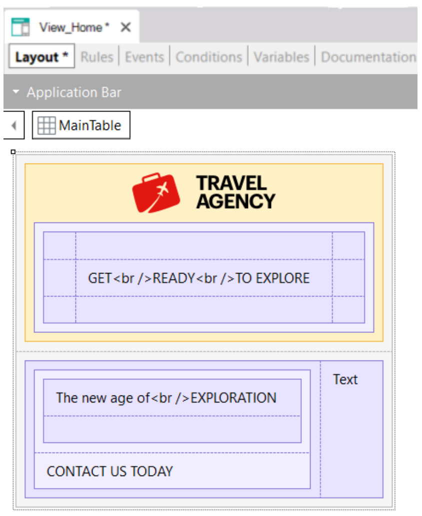
Now let's go to the variables section of MasterPanel_TravelAgency. We create a MenuItems variable, of the MenuItem data type, which is the variable based on the collection SDT.

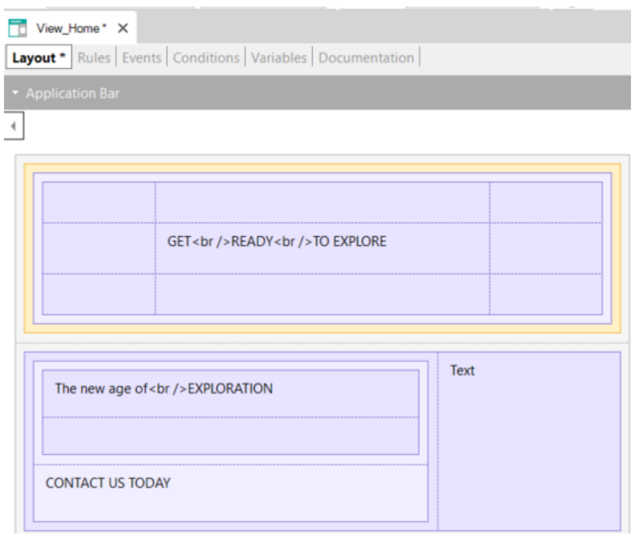
Then we create a MenuItem variable to which we will assign the MenuItem.items data type, where we will load each menu item to later add it to the items collection.

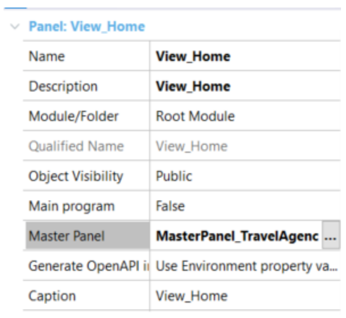
We open the events section of the panel and add a ClientStart event, where we will write the code needed to load our menu items collection.

Now go back to the panel’s layout and select the TravelAgencyMenubar1 user control. In its “items” property select the &MenuItems variable. This is informing the user control where it must search for the data to show the items.

Modify the startup panel object and assign the Master Panel


View_Home *
Layout * | Rules | Events | Conditions | Variables | Documentation
Application Bar
MainTable
TRAVEL AGENCY
GET<br />READY<br />TO EXPLORE
Text
The new age of<br />EXPLORATION
CONTACT US TODAY


View_Home *
Layout * | Rules | Events | Conditions | Variables | Documentation
Application Bar
GET<br />READY<br />TO EXPLORE
Text
The new age of<br />EXPLORATION
CONTACT US TODAY



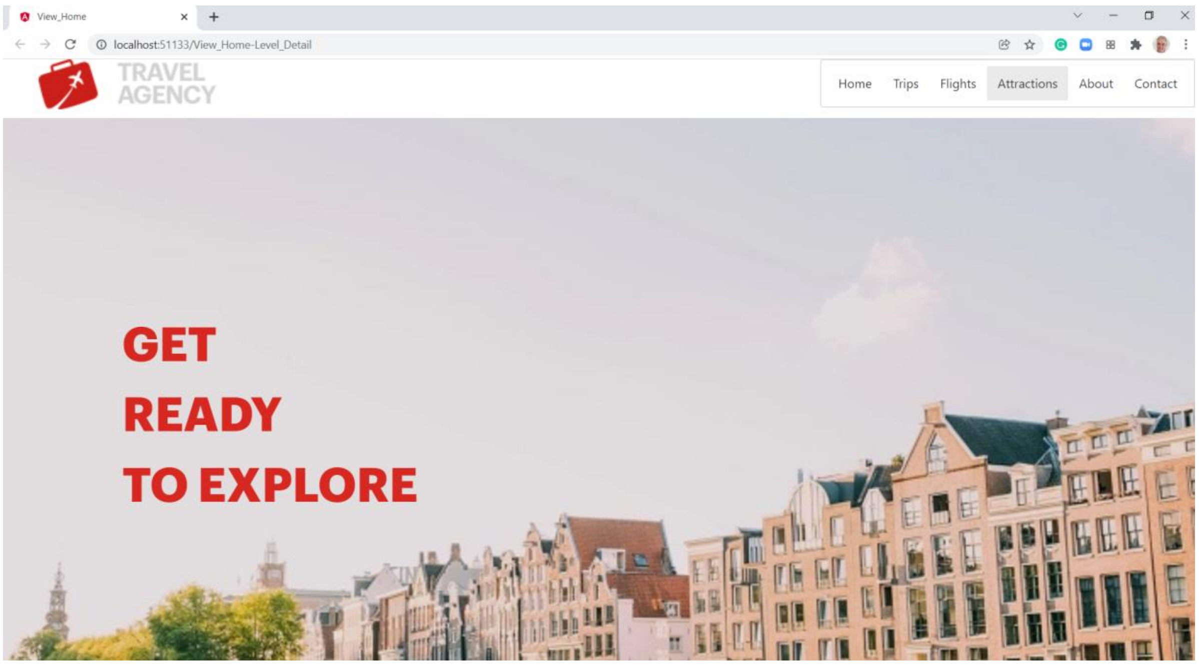
Panel: View_Home

Name	View_Home
Description	View_Home
Module/Folder	Root Module
Qualified Name	View_Home
Object Visibility	Public
Main program	False
Master Panel	MasterPanel_TravelAgenc ...
Generate OpenAPI i	Use Environment property va...
Caption	View_Home

To try this, open the View_Home panel that was the main object you created as startup object prior to using the design sent by the designer from Sketch, remove the icon and assign the Master Panel property with Master_Panel_TravelAgency, which already includes the icon and the menu.

Right click on the View_Home panel and select Run.

Executing the User Control



You will see the View_Home object open up, showing the menu on the screen's upper right.

When you drag the mouse over the buttons you will see them change in color as you go over them; this is to indicate the one that will be selected if you click.

You will not assign objects to the menu here because you already have the menu that you will actually use, that is, the one created from Sketch.

The use of user controls in an app generated in Angular implies certain considerations inherent to the framework's architecture, so you must refer to the documentation of each user control provider, or if you set up your own, then you should consider a way of including the modules, styles and other components necessary.

The possibility of defining your own user controls in GeneXus instead of having only the standard controls allows you to significantly increase the user experience of your apps, and makes it possible to use creations from a variety of sources and include them in your development.

In further videos you will also see how to include other external functionalities accessing APIs.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications