



Home



My Network



Jobs



Messaging



Notifications



Me



For Business

[Get hired with Premium](#)

Create your own newsletter

Start your own discussion with a newsletter on LinkedIn. Share your thought leadership with every new edition.

[Try it out](#)

SAP Spartacus: Customizing the Product Details Page Part 2

By Avinash Jadhav

SAP Spartacus: Customizing the Product Details Page Part 2



Avinash Jadhav

SAP Composable Storefront(formerly Spartacus) Trainer / Coach / Expert |...

5 articles

✓ Following

November 5, 2022

Open Immersive Reader

Hey, welcome back! Today I will show you how to customize the Product Details Page by SAP Spartacus library.

Note: Before moving into this article, please read [first part](#) of this article series on product details page customization

In the previous article we were learning how to identify CMS component which we need to customize and it was Product Details Page(PDP). In this article, I'm going to show you actual implementation with some small configurations.

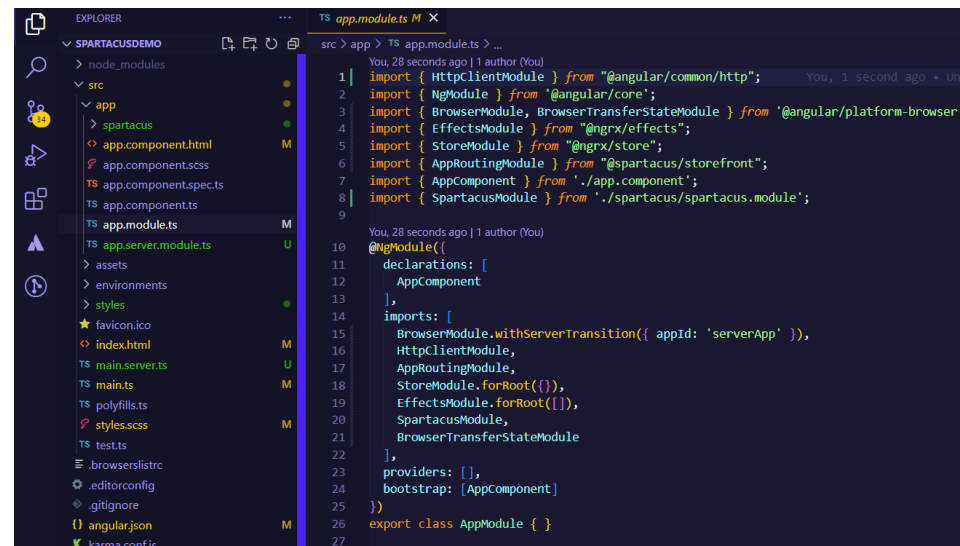
Previous article was finished with identifying CMS component which we are going to customize inside PDP page. Here is the last snippet of the previous article:

```
▼ 15: {slotId: "ProductSummarySlot",...}
  ▼ components: {component: [{uid: "ProductImagesComponent",...}, {uid: "ProductIntroComponent",...},...]}
    ▼ component: [{uid: "ProductImagesComponent",...}, {uid: "ProductIntroComponent",...},...]
      ► 0: {uid: "ProductImagesComponent",...}
      ▼ 1: {uid: "ProductIntroComponent",...}
        container: "false"
        flexType: "ProductIntroComponent"
        modifiedtime: "2022-08-21T20:58:28.621Z"
        name: "ProductIntroComponent"
        synchronizationBlocked: "false"
        typeCode: "CMSFlexComponent"
        uid: "ProductIntroComponent"
        uid: "eyJpdGVtSWQ1OjJQcm9kdWN0SW50cm9Db21wb251bnQiLCJjYXRhbG9nSWQ1OjJlbGVjdHJvbm1jcy1zcGFDb2"
      ► 2: {uid: "QualtricsEmbeddedFeedbackComponent",...}
      ► 3: {uid: "ProductSummaryComponent",...}
      ► 4: {uid: "VariantSelector",...}
      ► 5: {uid: "AddToCart",...}
      ► 6: {uid: "ConfigureProductComponent",...}
      ► 7: {uid: "AddToWishListComponent",...}
      ► 8: {uid: "StockNotificationComponent",...}
```

The CMS component which we are going to map is **ProductIntroComponent**

from **ProductSummarySlot**. From above screenshot we have to use property called **flexType** from ProductIntroComponent which is to be mapped with our custom angular component. We are going to use this **flexType** value in our Spartacus configuration.

Let me show the the current structure of my Spartacus Demo project.

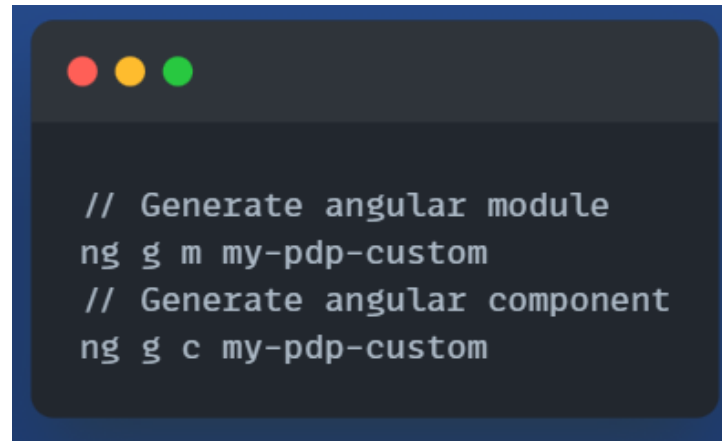


The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure for 'SPARTACUSDEMO'. The 'src' directory is expanded, showing 'app' and 'spartacus' subdirectories. The 'app' directory contains 'app.component.html', 'app.component.scss', 'app.component.spec.ts', 'app.component.ts', 'app.module.ts', 'app.server.module.ts', 'assets', 'environments', 'styles', 'favicon.ico', 'index.html', 'main.server.ts', 'main.ts', 'polyfills.ts', 'styles.scss', and 'test.ts'. The 'spartacus' directory contains 'spartacus.module.ts'. The main editor displays the 'app.module.ts' file, which contains the following code:

```
1 import { HttpClientModule } from '@angular/common/http';
2 import { NgModule } from '@angular/core';
3 import { BrowserModule, BrowserTransferStateModule } from '@angular/platform-browser';
4 import { EffectsModule } from '@ngrx/effects';
5 import { StoreModule } from '@ngrx/store';
6 import { AppRoutingModule } from '@spartacus/storefront';
7 import { AppComponent } from './app.component';
8 import { SpartacusModule } from './spartacus/spartacus.module';
9
10 @NgModule({
11   declarations: [
12     AppComponent
13   ],
14   imports: [
15     BrowserModule.withServerTransition({ appId: 'serverApp' }),
16     HttpClientModule,
17     AppRoutingModule,
18     StoreModule.forRoot([]),
19     EffectsModule.forRoot([]),
20     SpartacusModule,
21     BrowserTransferStateModule
22   ],
23   providers: [],
24   bootstrap: [AppComponent]
25 })
26 export class AppModule { }
27
```

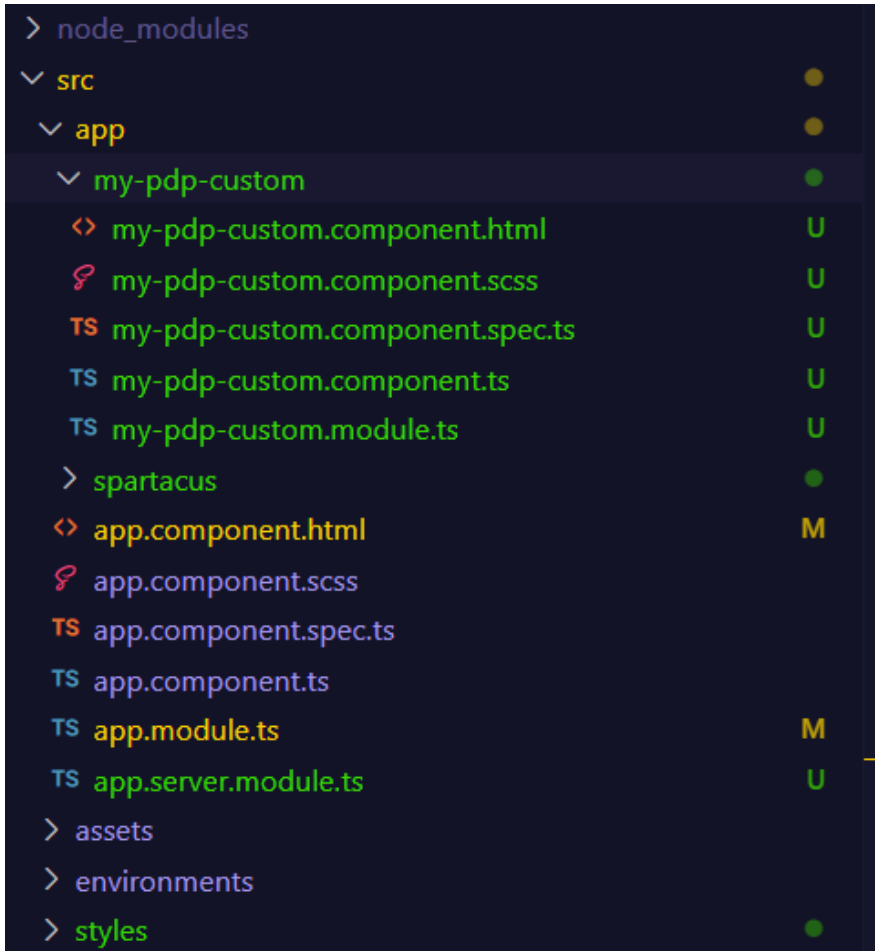
Now let's create our angular component and module which will be mapped with **ProductIntroComponent**.

Please use below commands to create angular component and module.

A terminal window with a dark blue border and a dark gray background. It features three colored window control buttons (red, yellow, green) in the top-left corner. The terminal contains two lines of text, each preceded by a comment: the first line is a comment followed by a command to generate an Angular module, and the second line is a comment followed by a command to generate an Angular component. Both commands use the 'ng g' syntax with the module name 'my-pdp-custom'.

```
// Generate angular module  
ng g m my-pdp-custom  
// Generate angular component  
ng g c my-pdp-custom
```

With
above commands my project structure will look something
like this.

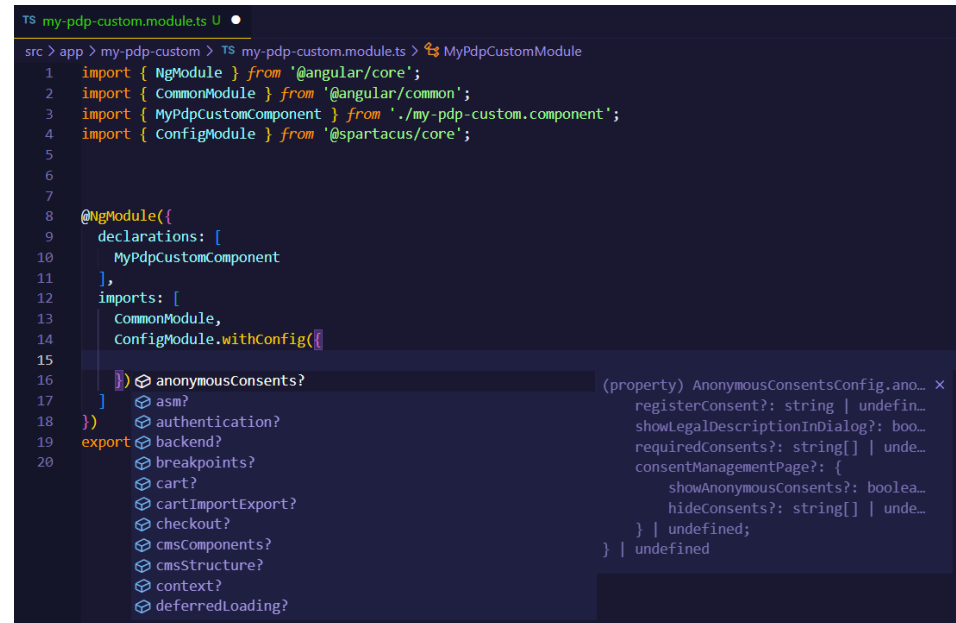


Now lets dive into actual configuration part of this article.

The file we are going to add our spartacus configuration are my-pdp-custom.module.ts

For adding spartacus configuration we have to use module called **ConfigModule** from **spartacus core** libraries. This

ConfigModule ships with method called **withConfig()**. In this method we have to pass our configuration.



```
TS my-pdp-custom.module.ts U
src > app > my-pdp-custom > TS my-pdp-custom.module.ts > MyPdpCustomModule
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { MyPdpCustomComponent } from './my-pdp-custom.component';
4 import { ConfigModule } from '@spartacus/core';
5
6
7
8 @NgModule({
9   declarations: [
10     MyPdpCustomComponent
11   ],
12   imports: [
13     CommonModule,
14     ConfigModule.withConfig({
15
16     })
17   ]
18 })
19 export {
20   anonymousConsents?
  (property) AnonymousConsentsConfig.ano... x
  registerConsent?: string | undefin...
  showLegalDescriptionInDialog?: boo...
  requiredConsents?: string[] | unde...
  consentManagementPage?: {
    showAnonymousConsents?: boolea...
    hideConsents?: string[] | unde...
  } | undefined;
  } | undefined
  asm?
  authentication?
  backend?
  breakpoints?
  cart?
  cartImportExport?
  checkout?
  cmsComponents?
  cmsStructure?
  context?
  deferredLoading?
```

If you look at above screenshot, you will get an idea of using **ConfigModule**. Inside withConfig method if you pass the object and if you press CTRL + Space, VScode will give you available configuration settings from spartacus. The one which we are going to use to map our component is **cmsComponent**.

The next screenshot will show you complete settings for mapping CMS component with angular component.

```

TS my-pdp-custom.module.ts U
src > app > my-pdp-custom > TS my-pdp-custom.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { MyPdpCustomComponent } from './my-pdp-custom.component';
4 import { ConfigModule } from '@spartacus/core';
5
6 @NgModule({
7   declarations: [
8     MyPdpCustomComponent
9   ],
10  imports: [
11    CommonModule,
12    ConfigModule.withConfig({
13      cmsComponents: {
14        ProductIntroComponent: {
15          component: MyPdpCustomComponent
16        }
17      }
18    })
19  ]
20 })
21 export class MyPdpCustomModule { }

```

Comopnent name coming from CMS as flexType

Custom component created by us.

The ProductIntroComponent is the name coming from CMS as **flexType** which we have used here in our configuration and as a value to that flexType component we have to provide object with name of your custom angular component.

One very small step I missed is to add cast to configuration object:

```

6  ∨ @NgModule({
7  ∨   declarations: [
8      MyPdpCustomComponent
9  ],
10 ∨   imports: [
11     CommonModule,
12     ConfigModule.withConfig({
13         cmsComponents: {
14             ProductIntroComponent: {
15                 component: MyPdpCustomComponent
16             }
17         }
18     } as CmsConfig) ←
19 ]
20 })
21 export class MyPdpCustomModule { }

```

Next step is to import your custom module inside app module of your application.

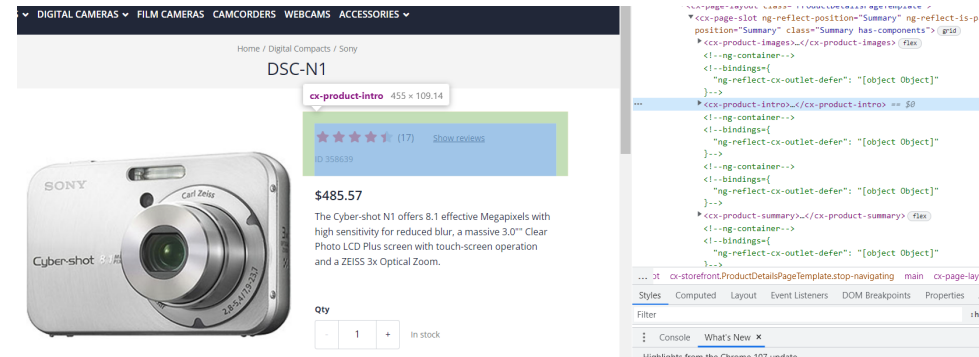
```

10  You, 4 minutes ago | 1 author (You)
11  @NgModule({
12      declarations: [
13          AppComponent
14      ],
15      imports: [
16          BrowserModule.withServerTransition({ appId: 'serverApp' }),
17          HttpClientModule,
18          AppRoutingModule,
19          StoreModule.forRoot({}),
20          EffectsModule.forRoot([]),
21          SpartacusModule,
22          BrowserTransferStateModule,
23          MyPdpCustomModule
24      ],
25      providers: [],
26      bootstrap: [AppComponent]
27  })
28  export class AppModule { }
29

```

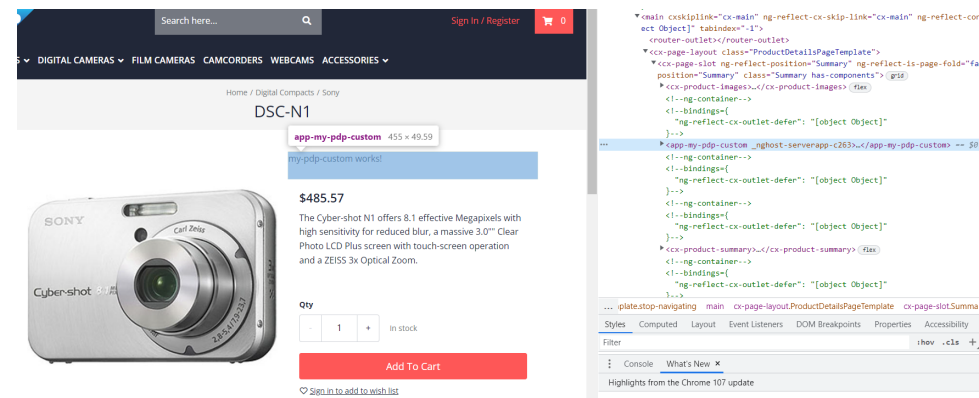

Below I have attached before and after adding the custom component.

Before:



You can see in the inspect section of developer console we have `cx-product-intro` which is coming from CMS.

After:



After successfully adding our custom component you can see that we have replaced spartacus out of the box ProductIntroComponent with our custom angular component. You can also see in the inspect section of developer tools where you can see **app-my-pdp-custom** which is our custom angular component we created in earlier steps.

And that's it guys for today! I hope you have learned a lot from this second part of my series on Customizing PDP component.

Please share this article with your friends, colleagues, team mates and help them too.

Sharing will motivate me to write more and more on this topic! Cheers!

Report this

Published by



Avinash Jadhav

SAP Composable Storefront(formerly Spartacus) Trainer / Coach / Expert | Ang...
Published • 1y

5

articles

✓ Following

Hi [#connections](#) , 2nd part of my article series SAP Spartacus : Product details page customization is available now. [#sap](#) [#spartacus](#) [#sapcommercecloud](#)

#sapcommerce #storefront #sapcommunity #headless #headlesscommerce
#vuestorefront

 Like  Comment  Share

 aditya yadav and 3 others 1 comment

Reactions



1 Comment



Add a comment...



aditya yadav (He/Him) • 1st
Technical Lead

1y ...

Great 👍

Like | Reply




Avinash Jadhav

SAP Composable Storefront(formerly Spartacus) Trainer / Coach / Expert | Angular |
React.js

✓ Following


More from Avinash Jadhav



SAP Spartacus
Customizing Your Store with LayoutConfig

Customizing Your Store with LayoutConfig in SAP Spartacus

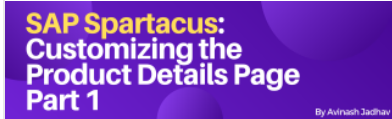
Avinash Jadhav on LinkedIn



SAP Spartacus
The Key to Greater Success as an Angular Developer

SAP Spartacus: The Key to Greater Success as an Angular Developer

Avinash Jadhav on LinkedIn



SAP Spartacus:
Customizing the Product Details Page
Part 1

By Avinash Jadhav

SAP Spartacus: Customizing the Product Details Page Part 1

Avinash Jadhav on LinkedIn

[See all 5 articles](#)

[About](#)

[Community Guidelines](#)

[Privacy & Terms](#) ▼

[Sales Solutions](#)

[Safety Center](#)

[Accessibility](#)

[Careers](#)

[Ad Choices](#)

[Mobile](#)

[Talent Solutions](#)

[Marketing Solutions](#)

[Advertising](#)

[Small Business](#)



Questions?

Visit our Help Center.



Manage your account and privacy

Go to your Settings.



Recommendation transparency

Learn more about Recommended Content.

Select Language

English (English)

