WCMS
Exercise

SAP Customer Experience

TABLE OF CONTENTS

GOAL

This exercise covers the creating of a new frontend component (an annotated banner) and its associated view.

We're going to display this new special banner on the homepage. Such a goodie should be available only to our best customers, so only logged-in users will see it.

INSTRUCTIONS

Preparation

P1 Begin by making sure the server is stopped, then invoke the exercise's **setup** ant target:

P1.1 If you haven't done so in the current *Terminal* or *cmd* window, set Ant home environment variables by navigating to the `MYPATH/workspace/hybris/bin/platform` directory and executing:

`. ./setantenv.sh` (on MacOS or Linux) or `setantenv.bat` (on Windows).

P1.2 Then navigate to `MYPATH/workspace/TrainingLabTools/exercise_WCMS` and execute:

```
ant -f wcms_tasks.xml setup
```

This task creates the **trainingwcms** extension, where you will perform some of your work. It also adds the extension to your `localextensions.xml` file.

Don't forget to import the **trainingwcms** extension into eclipse.

Step 1 - Define an SAP Commerce Item Type for the new CMS Component

Before we can create an instance of an annotated banner component in *SmartEdit*, it needs to be defined as a type. For simplicity's sake, we'll just extend the existing **SimpleResponsiveBannerComponent** type. An instance of that type is just a database entity – it doesn't provide functionality.

So… you want to create a new item type for our annotated banner component.

1.1 To do so, edit *MYPATH*`/workspace/hybris/bin/custom/trainingwcms/resources/trainingwcms-items.xml`, and add the following code inside the `<itemtypes>` tag:

```
<itemtype code="AnnotatedResponsiveBannerComponent"
          extends="SimpleResponsiveBannerComponent"
          autocreate="true"
          generate="true">
    <attributes>
        <attribute type="java.lang.String" qualifier="title">
            <persistence type="property"></persistence>
```

```
            </attribute>
        </attributes>
    </itemtype>
```

Since our new item type extends an out-of-the-box one, we need the compiler to see it.

1.2 The file `MYPATH/workspace/hybris/bin/custom/trainingwcms/extensioninfo.xml` includes a list of required extensions. Replace the comment you'll find there with the following:

```
<requires-extension name="acceleratorcms"/>
```

We also want to localize it properly.

1.3 Add the content below to the localization file `MYPATH/workspace/hybris/bin/custom/trainingwcms/resources/localization/trainingwcms-locales_en.properties`:

```
type.AnnotatedResponsiveBannerComponent.name=Annotated Responsive Banner

type.AnnotatedResponsiveBannerComponent.description=Annotated Responsive Banner for CX Commerce Education

type.AnnotatedResponsiveBannerComponent.title.name=Title

type.AnnotatedResponsiveBannerComponent.title.description=Title of the Annotated Responsive Banner
```

1.4 Perform an `ant all`, which generates the model classes for you,
Then start the server. Go to HAC, open Platform | Update, then check

- *Update running system*
- *Localize Types*

and perform the update. After the update is finished, keep the server running and continue with the next step.

Step 2 - Develop the Rendering Logic and View

Now we can implement an Angular component to relate to the new cms component type, which will use the data passed from SAP Commerce Cloud to render the corresponding view.

2.1 As a best practice, let's first create a new Angular module.

Please navigate to the Composable Storefront application root directory (e.g., mystore or mystore500 from the composable storefront exercise. We will use `MYSTOREPATH` to refer to the root directory in this document from now on), then execute:

```
ng g module spartacus/features/exercise-cmscomponents
```

This will create a new folder inside the `MYSTOREPATH/src/app/spartacus/features/` folder.

Since we are going to use the new module in the composable storefront code base, let's also edit the

`MYSTOREPATH/src/app/spartacus/features/spartacus-features.module.ts`

and include the `exercise-cmscomponents` module as below:

```
import { NgModule } from '@angular/core';
...
import { AsmFeatureModule } from './features/asm/asm-feature.module';
import { ExerciseCmscomponentsModule } from './features/exercise-cmscomponents/exercise-cmscomponents.module';

@NgModule({
  declarations: [],
  imports: [
    ...
    AsmFeatureModule,
    ExerciseCmscomponentsModule
  ]
})
export class SpartacusFeaturesModule { }
```

Please note (cf. above), you only need to

- import the `exercise-cmscomponents` module by specifying the location of the module,
- and declare it in the NgModule list.

2.2 Now, you can create a new Angular component inside the `exercise-cmscomponents` with the following command (still while in the MYSTOREPATH directory):

`ng g component spartacus/features/exercise-cmscomponents/annotated-banner`

This will be the new ***AnnotatedBannerComponent*** responsible for rendering the view of the ***AnnotatedResponsiveBannerComponent*** (defined earlier in step 1).

Open `MYSTOREPATH/src/app/spartacus/features/exercise-cmscomponents/annotated-banner/annotated-banner.component.ts` , implement *AnnotatedBannerComponent* as follows:

```
import { Component, OnInit } from '@angular/core';
import { Image, CmsBannerComponent, ImageGroup } from '@spartacus/core';
import { CmsComponentData } from '@spartacus/storefront';
import { Observable } from 'rxjs';
```

```
export interface AnnotatedCmsBannerComponent extends CmsBannerComponent {
  title?: string;
}

@Component({
  selector: 'app-annotated-banner',
  templateUrl: './annotated-banner.component.html',
  styleUrls: ['./annotated-banner.component.scss']
})
export class AnnotatedBannerComponent implements OnInit {

  data$: Observable<AnnotatedCmsBannerComponent> = this.componentData.data$;

  constructor(private componentData: CmsComponentData<AnnotatedCmsBannerComponent>) { }

  ngOnInit(): void {
  }

  getImage(data: AnnotatedCmsBannerComponent): Image | ImageGroup | undefined {
    if (data.media) {
      if ('url' in data.media) {
        return data.media as Image;
      } else {
        return data.media as ImageGroup;
      }
    }
    return undefined;
  }


}
```

2.3 Open `MYSTOREPATH/src/app/spartacus/features/exercise-cmscomponents/annotated-banner/annotated-banner.component.html` , implement view as follows:

```
<ng-container *ngIf="data$ | async as data">
    <h1>{{data.title}}</h1>
    <cx-media [container]="getImage(data)"></cx-media>
  </ng-container>
```

We'll use the standard **cx-media** composable storefront component to render the original image of the banner component, and also use HTML <h1> to define the banner heading based on the customized banner component's title property.

2.4 Map this Angular component to the new CMS component type.

Open `MYSTOREPATH/src/app/spartacus/features/exercise-cmscomponents/exercise-cmscomponents.module.ts`, change the code as follows:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AnnotatedBannerComponent } from './annotated-banner/annotated-
banner.component';
import { MediaModule } from '@spartacus/storefront';
import { CmsConfig, ConfigModule } from '@spartacus/core';

@NgModule({
  declarations: [
    AnnotatedBannerComponent
  ],
  imports: [
    CommonModule,
    MediaModule,
    ConfigModule.withConfig({
      cmsComponents: {
        AnnotatedResponsiveBannerComponent: {
          component: AnnotatedBannerComponent,
        },
      },
    } as CmsConfig)
  ],
  exports: [AnnotatedBannerComponent]
})
export class ExerciseCmscomponentsModule { }
```

It means whenever the Composable Storefront gets CMS component data for the ***AnnotatedResponsiveBannerComponent*** from SAP Commerce Cloud, the ***AnnotatedBannerComponent*** will be used to provide the rendering logic and view.

Go to the root directory of Composable Storefront and start it with the following command in the Terminal window:

```
yarn start --ssl
```

Step 3 - Authorize the new CMS Component Type

We need to tell SAP Commerce Cloud that the new component type is valid for the **homepage slot**. In our WCMS model, that should be done through a *Component Type Group*.

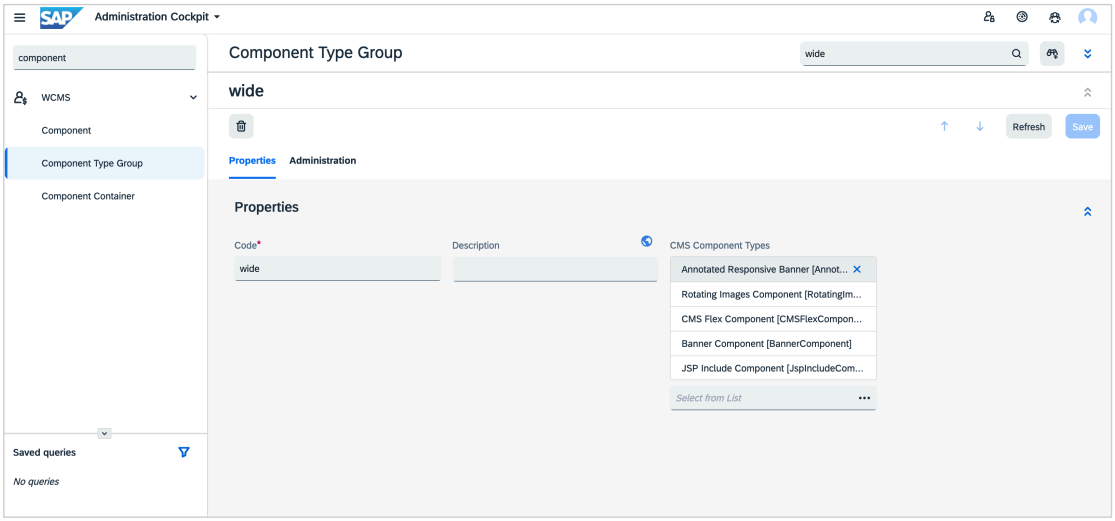
Every slot (or location) in a page template can be associated with a Component Type Group that lists all the WCMS component types that are allowed to be placed in that slot.

We want to put our new component type in **Section1** of the *homepage*. First, let's find out what Component Type group is associated with this slot.

3.1 Navigate to Backoffice and select *WCMS* > *Page*. Locate the ***homepage*** CMSPage (in the SPA Electronics Content Catalog: Staged) and open it for editing. On the first tab, locate its **Page Template**. Double-click on the value there (**Landing Page 2 Template**), and a pop-up window that appears showing the template's properties. In the list that is the **Available Content Slots** property, scroll down until you see **Section1**; double-click that entry to display its **Component Type Group** property. It is the **wide** Component Type Group. You can close both pop-up windows now.

Armed with that knowledge, we know we need to add our new component type to the **wide** Component Type Group before we can add it to the **Section1** slot on the **Homepage** in SmartEdit.

3.2 In Backoffice, select *WCMS* > *Component Type Group*, scroll down and open the item named **wide**. In the first tab (**PROPERTIES**), a list of CMS Component Types appears. In the search box at the bottom of that list, search for ***AnnotatedResponsiveBannerComponent***, then click on the item displayed to add it to the list. Remember to click **SAVE**.



Step 4 - Add the new CMS Component to the Homepage

3.1 In a browser, open SmartEdit at https://localhost:9002/smartedit/. Log in with administrator credentials (admin/nimda).

Under the Site, make sure to choose the Spartacus Electronics Site and note the 2 content catalog versions: Staged and Online.

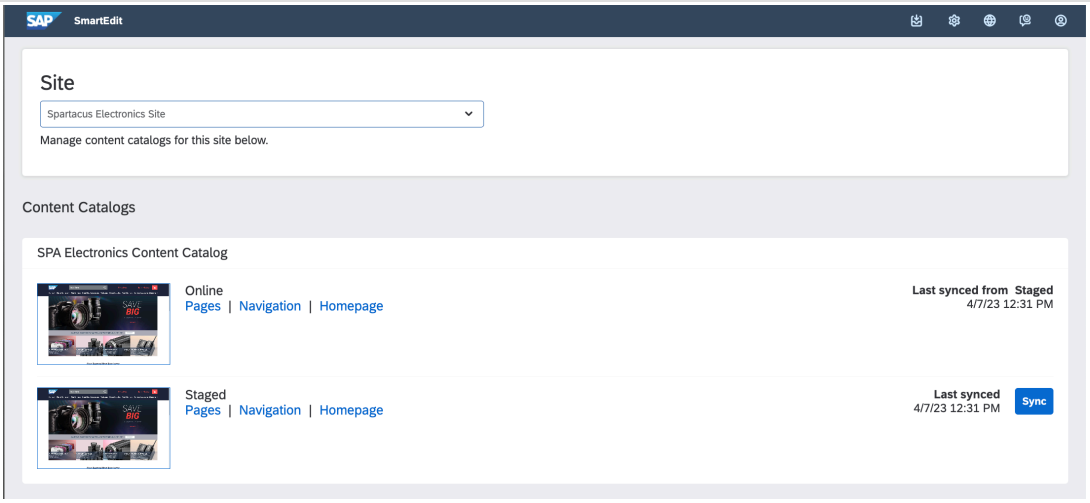


Figure: Sites and content catalogs in SmartEdit

3.2 Click the **Homepage** link in the Staged catalog version.

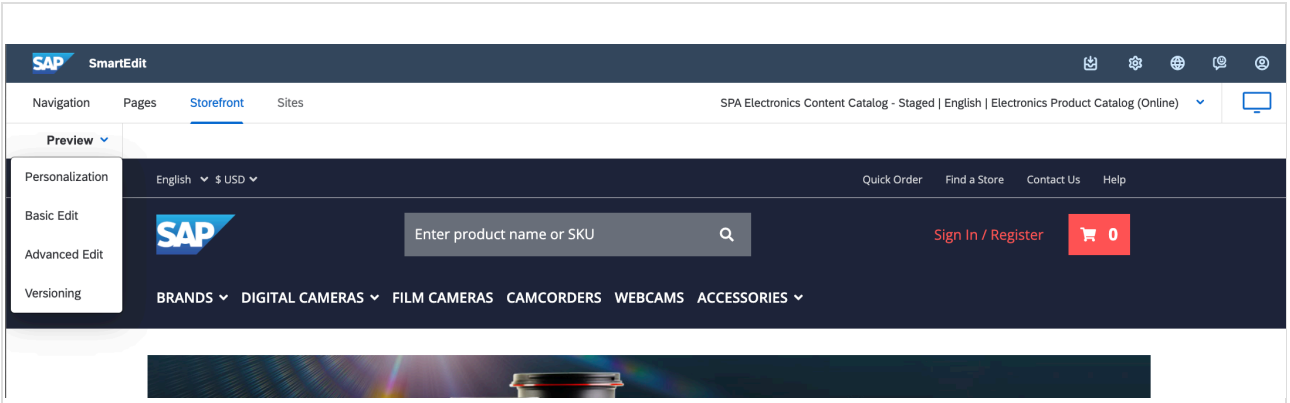


Figure: SmartEdit Modes

3.3 In the top-left corner, click the mode selector drop-down (it might show **Preview**, assuming this is your first time accessing SmartEdit), and select **Advanced Edit**.

3.4 Click **Page Structure** at the top of the page to open the structured tree of the current page including content slots and components. Locate the Section1Slot-Homepage and click on it, it should contain 2 responsive banner components by default.

Then click on the **+ Component** link next to **Page Structure**:

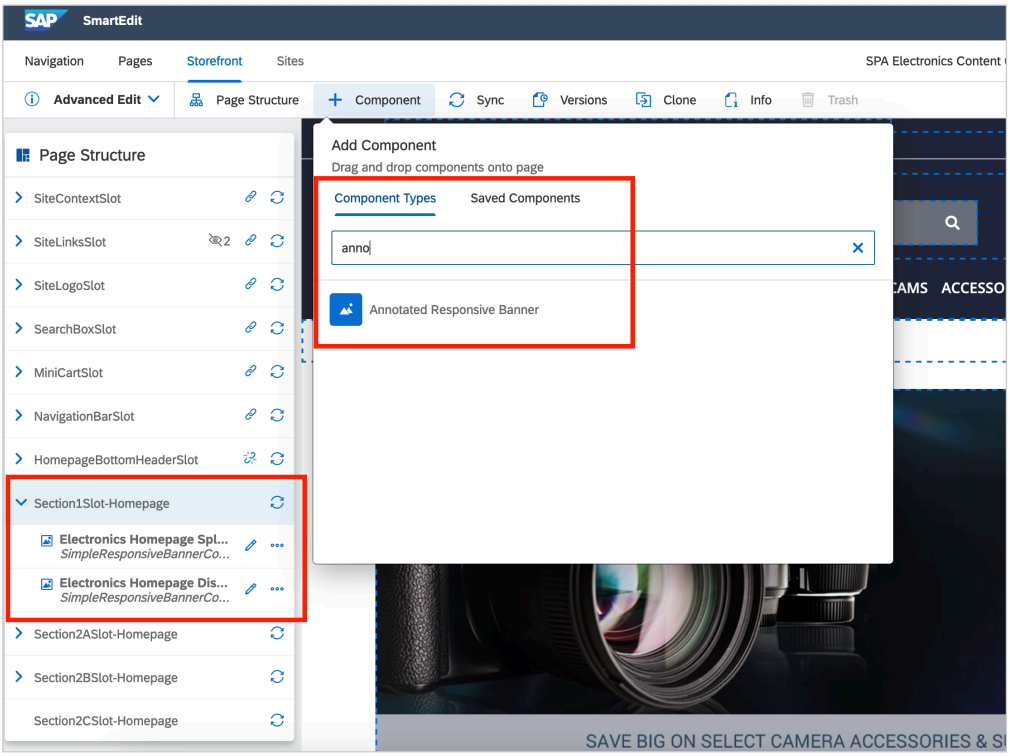


Figure: Add Component Drop-Down

3.5 In the drop-down that appears (cf. above), type "anno" under "Component Types" and select the **Annotated Responsive Banner** component, and drag it to the top of the Section1 slot (Section1Slot-Homepage) in the Page Structure view, just above the **Electronics Homepage Splash Banner** component (the first component in the section1).

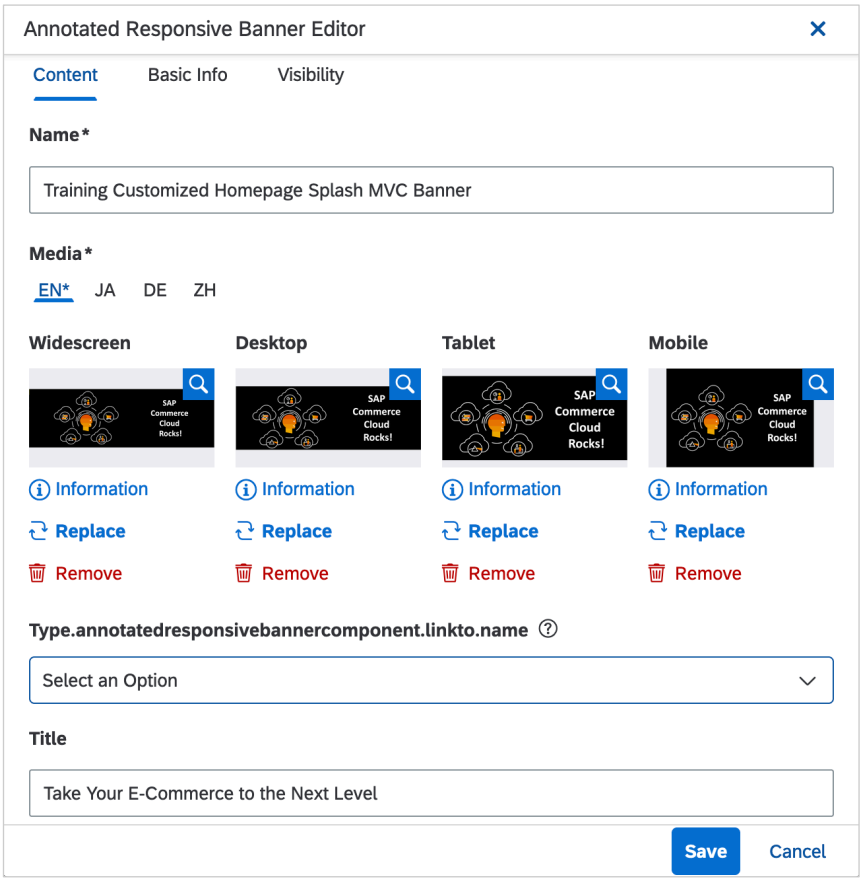


Figure: Drop New Component on Section1 Slot above the first Banner component

3.6 In the editor that appears (cf. above):

- type "**Training Customized Homepage Splash MVC Banner**" for the **Name**,
- for the **Media** property, you can click on the Upload button and navigate to the newly created extension trainingwcms's folder, choose the subfolder resources/media, and use:
  - WCMSExercise_Widescreen.png for Widescreen
  - WCMSExercise_Desktop.png for Desktop
  - WCMSExercise_Tablet.png for Tablet
  - WCMSExercise_Mobile.png for Mobile

  Respectively. And **remember** to click on the Upload button **each time** after select the corresponding image file, as this can be overlooked easily.
- type "**Take Your E-Commerce to the Next Level**" for the **Title** of the component (or anything else you want to use for the annotated banner).

Finally, save the Annotated Banner component.

3.7 Verify that your component appears correctly. If it does, click on the downward arrow next to **Draft** (in the top right corner, cf. the screenshot below) and select **Ready To Sync**.

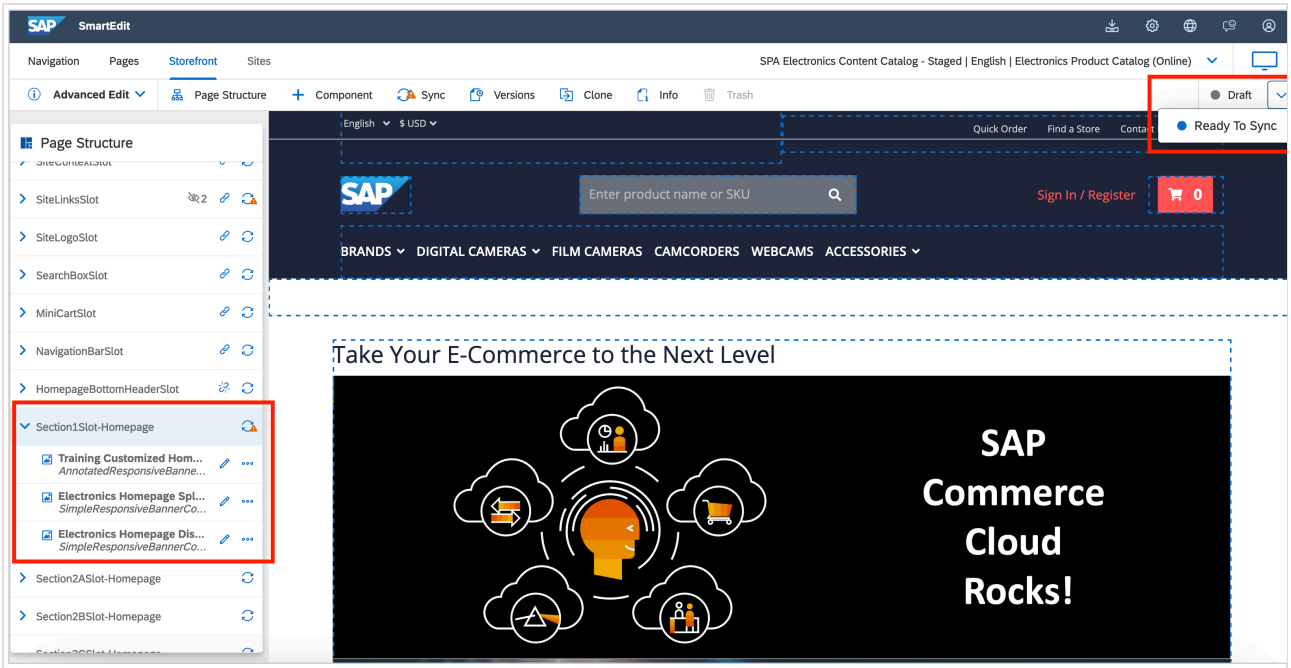
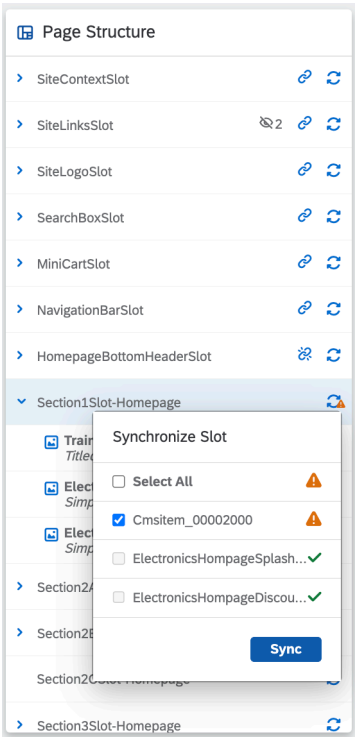


Figure: New Component Added to Section 1 Slot is Ready to Sync

3.8 Under **Page Structure**, Click on the **Sync** button of **Section1Slot-Homepage**, select the new component in the dropdown list provided, then click on the **Sync** button.



3.9 View the online storefront at https://localhost:4200/USD/en/electronics-spa/, and check that your new component appears on the Homepage.

Step 4 - Add a CMS Component Restriction

4.1 Return to SmartEdit, expand the Section1Slot-Homepage in the Page Structure view and click on the pencil icon that appears for this new component **Training Customized Homepage Splash MVC Banner**.

In the dialog that appears switch to the **Visibility** tab. There, let's **Create** a restriction.

In the **Add Restriction** pane that appears, select **Inverse Restriction** from the **Restriction Type** drop-down, and **Logged in User** as the **Restriction** (your only choice). SmartEdit tells you this is the inverse of the **Anonymous User Restriction** (which checks whether the user is anonymous or not). Make sense, right? Click **Add**, then **Save**.

Notice that the component disappears, **Section1Slot-Homepage** now has a crossed-out eye icon with number 1 indicating that it contains one WCMS component with restriction. (Note: if you can't see the crossed-out eye icon yet, try to refresh the page)

With the **Section1Slot-Homepage** expanded, the **Training Customized Homepage Splash MVC Banner** component is displayed with a crossed-out eye icon as well.

4.2 Set the page to **Ready To Sync**, then synchronize it using the **Sync** button on the top (either the entire page or you can also just sync the cms component in section1 directly).

4.3 View the online website at https://localhost:4200/USD/en/electronics-spa/, and check that the component no longer appears.

4.4 Now you can register a new customer account.

Please note: when registering a new customer, you need to give a password fulfilling the default composable storefront password policy, e.g., Test_123).

After logging into the storefront, the annotated banner component should reappear on the home page.

Verify

To verify your solution, go to the HAC and run the script `verifyWCMSExercise`.

Solution

If you don't wish to complete this exercise manually, you can run the solution provided:

S1 Navigate to the *Terminal* or *cmd* window where the server is running, and if it is running, stop it by entering `CTRL-C`.

S2 Navigate to *`MYPATH`*`/workspace/TrainingLabTools/exercise_WCMS` and execute:

```
ant -DcomposableStorefrontRoot=#ROOT-OF-COMPOSABLE_STOREFRONT# -f wcms_tasks.xml
solution
```

Here, `ROOT-OF-COMPOSABLE_STOREFRONT#` refers to the directory of the Composable Storefront application (e.g., mystore or mystore500).

After the ant command successfully ends, please start the commerce server and synchronize the electronics-spaContentCatalog staged version to the electronics-spaContentCatalog online before you perform any test.

RECAP

In this exercise, you learned how to create a new CMS component type, develop a related Angular component to provide the rendering logic and view, authorize the CMS component type and add it to the composable storefront's home page. You also learned how to restrict a cms component from appearing to certain users (anonymous users in this case).

www.sap.com

© 2023 SAP SE or an SAP affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See www.sap.com/copyright for additional trademark information and notices.

SAP Customer Experience