



# Spartacus Training

Day 3

---

# Recap.

---

We have added Spartacus into our existing angular application.

We learnt about lifecycle hooks.

*Now in this session we are going to learn on how to customize the looks and content of the Spartacus application.*



# Outlets

---

Outlets are the template driven approach to modify the specific content or layout.  
Specifically. TemplateRefs can be added using ng-template

Like

```
<ng-template cxOutletRef="CX_OUTLET_REF_NAME">  
  <!-- Your custom UI CODE -->  
</ng-template>
```

# Outlets



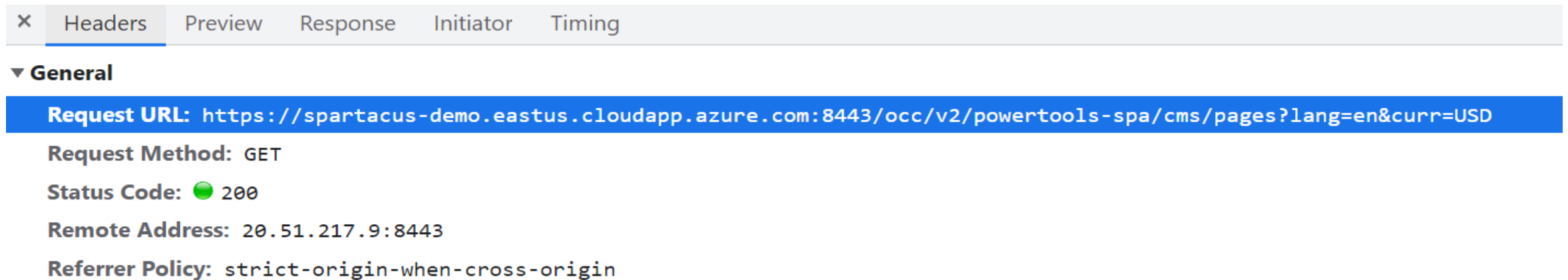
```
<body data-new-gr-c-s-check-loaded="14.1036.0" data-gr-ext-installed cz-shortcut-listen="true">
  <app-root _ngghost-dxi-c297 ng-version="10.2.5">
    <cx-storefront _ngcontent-dxi-c297 tabindex="0" class="LandingPage2Template stop-navigating">
      ::before
      <cx-skip-link>...</cx-skip-link>
      <header cxskiplink="cx-header" ng-reflect-cx-skip-link="cx-header" ng-reflect-config="[object Object]" tabindex="-1">
        <cx-page-layout section="header" ng-reflect-section="header" class="header">
          <cx-page-slot ng-reflect-position="PreHeader" ng-reflect-is-page-fold="false" position="PreHeader" class="PreHeader has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="SiteContext" ng-reflect-is-page-fold="false" position="SiteContext" class="SiteContext has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="SiteLinks" ng-reflect-is-page-fold="false" position="SiteLinks" class="SiteLinks has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="SiteLogo" ng-reflect-is-page-fold="false" position="SiteLogo" class="SiteLogo has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="SearchBox" ng-reflect-is-page-fold="false" position="SearchBox" class="SearchBox has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="SiteLogin" ng-reflect-is-page-fold="false" position="SiteLogin" class="SiteLogin has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="MiniCart" ng-reflect-is-page-fold="false" position="MiniCart" class="MiniCart has-components">...</cx-page-slot>
          <cx-page-slot ng-reflect-position="NavigationBar" ng-reflect-is-page-fold="false" position="NavigationBar" class="NavigationBar has-components">...</cx-page-slot>
        </cx-page-layout>
      </header>
    </cx-storefront>
  </app-root>
</body>
```

Let's change it for header. As you can see here, we are having layout of angular-Spartacus application.

Let's understand this.

# Outlets

Hybris controls the content and layout of the storefront! Why did I say that??



× Headers Preview Response Initiator Timing

▼ General

**Request URL:** `https://spartacus-demo.eastus.cloudapp.azure.com:8443/occ/v2/powertools-spa/cms/pages?lang=en&curr=USD`

**Request Method:** GET

**Status Code:** ● 200

**Remote Address:** 20.51.217.9:8443

**Referrer Policy:** strict-origin-when-cross-origin

Every time a URL changes this API gets called. This API in turn will return us the layout data of whole page.

# Outlets

```

X Headers Preview Response Initiator Timing
▼ {uid: "homepage",...}
  ▼ contentSlots: {contentSlot: [{slotId: "Section1Slot-Homepage",...}, {slotId: "Section2ASlot-Homepage",...},...]}
    ▼ contentSlot: [{slotId: "Section1Slot-Homepage",...}, {slotId: "Section2ASlot-Homepage",...},...]
      ▶ 0: {slotId: "Section1Slot-Homepage",...}
      ▶ 1: {slotId: "Section2ASlot-Homepage",...}
      ▶ 2: {slotId: "Section2BSlot-Homepage",...}
      ▶ 3: {slotId: "Section3Slot-Homepage",...}
      ▶ 4: {slotId: "Section4Slot-Homepage",...}
      ▶ 5: {slotId: "Section5Slot-Homepage",...}
      ▶ 6: {slotId: "HomepageBottomHeaderSlot",...}
      ▶ 7: {slotId: "SiteLogoSlot",...}
      ▶ 8: {slotId: "HomepageNavLinkSlot",...}
      ▶ 9: {slotId: "NavigationBarSlot",...}
      ▶ 10: {slotId: "MiniCartSlot",...}
      ▶ 11: {slotId: "FooterSlot",...}
      ▶ 12: {slotId: "HeaderLinksSlot",...}
      ▶ 13: {slotId: "SearchBoxSlot",...}
      ▶ 14: {slotId: "TopHeaderSlot",...}
      ▶ 15: {slotId: "PlaceholderContentSlot",...}
      ▶ 16: {slotId: "SiteContextSlot",...}
      ▶ 17: {slotId: "SiteLinksSlot",...}
    label: "homepage"

```



# Outlets

---

```
const factory = this.componentFactoryResolver.resolveComponentFactory(yourComponent)
this.outletService.add('cx-storefront', factory, OutletPosition.BEFORE)
```

It's an Alternative way to use outlets. Using dependency injections.

# Exercise

---

1. Show hide outlet on a button click(toggle functionality).
2. Add a like functionality on PDP page.
3. Change Search icon
4. Change Site logo.
5. <https://sap.github.io/spartacus-docs/outlets/> Use let-<var\_name> and get the data in

section1 slot <pre>{{model.component\$ | async | json}}</pre>





The image features a dark blue background with four decorative geometric patterns in the corners. The top-right corner has a teal diamond grid pattern. The bottom-left corner has a teal stepped line pattern. The bottom-right corner has a teal zigzag line pattern. The top-left corner is empty. The text "Thank you !" is centered in a bold, orange-brown font.

**Thank you !**

---