This chapter discusses customer experience management in SAP Commerce. It also covers the suite’s tools for personalization, including context-driven services, the personalization and conversion optimization solution that lives alongside these core management tools.

“Customer Experience Management”

Table of Contents

Index

The Authors

Seema Vishnoi, Sietse Verhoog, Drew Messinger-Michaels

SAP Commerce Cloud: Commerce with SAP C/4HANA

531 Pages, 2018, $89.95
ISBN 978-1-4932-1699-4

www.sap-press.com/4621
Chapter 5
Customer Experience Management

From the moment that an end customer arrives at your storefront, every moment that he interacts with what you’ve built will fall to some degree under customer experience, the topic of this chapter. What do you want to enable him to do, and how will the experience look and feel? How will you encourage him to return?

Customer experience (CX) refers not only to the design and flow of a customer-facing web frontend—what we generally call user experience (UX)—but also to the way that experience extends beyond any one channel and to the features and functions that enable that omnichannel experience. A given customer-facing user interface is simply the frontend toolset with which your customers will interact. Customer experience includes that UI but also incorporates the functional flow from one UI or channel to another, the data and media strategies that support those interfaces, and, most importantly, the broader philosophies and principles that drive the design. Everything that affects your customers’ interactions with what you’ve built, from the moment you begin designing it, falls within the purview of customer experience.

When we talk about managing customer experience within SAP Commerce, some of that more abstract philosophical work has of course been done for you. As we will discuss, there are omnichannel strategies built into the platform, as well as a set of features and functions that will allow you to innovate beyond the current standard of omnichannel and, in some cases, beyond the boundaries of standard commerce.

SAP Commerce draws a distinction between product content management (which we discussed in the previous chapter) on the one hand and customer experience management on the other. Data, media, design, and rules directly related to products or categories belong to your website’s product catalog, whereas the rest of your content, from pages and page components to any media that is independent of structured product data, belongs to your content catalog, which is versioned much like the product catalog (with staged and online versions) but handled separately.
In this chapter, we’ll discuss the design and management of content catalogs in SAP Commerce, as well as the suite’s tools for personalization, including the newest addition to customer experience in SAP Commerce, context-driven services, the robust personalization and conversion optimization solution that lives alongside these core management tools.

5.1 Page Templates for All Channels

SAP Commerce uses responsive technology to deploy to multiple devices from a single set of page elements. This means that rather than detecting the browser or operating system that the customer is using and then serving up a page template accordingly (an approach called adaptive design), SAP Commerce accelerator storefronts detect the number of pixels being displayed and then select and size their content accordingly.

Note

If you ever want to see whether your favorite site—or, for that matter, your least favorite site—uses responsive design, just resize your browser window; if the layout of the page changes, then you’re looking at something responsive.

The pixel counts are typically divided into breakpoints, which allow the responsive frontend to determine whether it is likely displaying for a desktop/laptop, a tablet, a mobile device, and so on. In principle, this approach can extend to any number of devices, if some new screen size standard comes into play (perhaps, say, as wearable technology continues to develop).

As an example, Figure 5.1 shows a block of content as displayed on the desktop breakpoint of the responsive homepage in the apparel store of the SAP Commerce accelerators.

Figure 5.2, by way of contrast, is part of the same set of content, resized for a mobile device breakpoint.
Here we see two different approaches to adjusting content for different devices. In the case of the Shop Women banner, the image has simply been resized, making it more prominent in the more vertically oriented mobile layout. The Start Your Season banner, however, is a different crop of the source image, which preserves the model as the focal point of the image, as well as the slogan and the call to action, in a way that simply resizing a full-width banner intended for desktop screens would not.

It is up to an individual content team to decide which content should be resized and which should be swapped out for different imagery entirely. In general, you should keep the horizontality of mobile screens in mind and avoid visuals or elements or (especially) text or buttons that will be difficult to interpret when displayed at a smaller size.

In addition, any design elements that require the user to hover over an image should be eliminated for mobile and tablet designs because there is effectively no way to hover over an element on a touch screen. Indeed, this fact means that hover-overs have fallen out of favor in general for SAP Commerce accelerator storefronts. Customers generally want a consistent experience across channels in terms of the basic actions that are possible through the user interface.

So if customers on mobile devices are not going to be able to hover over an image of the front of a dress to see the back of it, for example, then you may not want to set that convention on your desktop site. This is not to say that hover-over is outside of best practices in and of itself, of course. It’s simply a question of starting from what your customers expect on mobile and then fostering consistency. This point of caution ties into the larger principle of mobile-first design. The earliest mobile web pages essentially took existing desktop pages and shrunk down as many elements as possible to fit on a smaller screen—which, from a design philosophy perspective, is precisely backwards. That approach can easily result in page elements that are unwieldy, unclear, or otherwise unoptimized.

To avoid that, a mobile-first approach reorients your thinking in just the way that the term implies, by challenging you to first consider the smallest screen on which your content will ever be displayed, deciding which page elements are essential in that context, and then building out from there.

**Searching and Browsing**

Optimized mobile design is beneficial throughout SAP Commerce experiences, but it’s more important in some areas of your storefront than in others. Some customers, especially older customers, tend to be reluctant to convert while using a mobile device—but these customers do tend to search, research, learn, select, and even build shopping carts or builds of materials on mobile devices. This means that it’s tremendously important to let such customers continue their purchasing processes on other devices, as discussed in Chapter 3, Section 3.2. It also means that mobile-optimized content, and therefore mobile-first design, is even more important on product detail pages, product listing pages, and content pages than it is for checkout or account management pages. Even if you have a complex purchasing process that cannot be entirely optimized for mobile, it is very much in your interest to optimize the search and browsing experience for mobile.

Elements that are vital to a customer’s initial interactions with your storefront should be initially visible upon arriving at the storefront. Elements that are necessary but may not be immediately necessary, such as account management tools, are typically hidden behind an icon composed of three or four horizontal lines (usually called a hamburger) on the mobile site, as shown on the left in Figure 5.3.

**Figure 5.3 Compressed Mobile UI Elements, Including Hamburger, in the Electronics Storefront**

The responsive technology in SAP Commerce accelerator storefronts works by compiling Leaner Style Sheets (LESS) into standard Cascading Style Sheets (CSS) for each breakpoint-specific layout. The system also makes use of resource minification, meaning that in the cases of media and code and scripts, the system will try to reduce the size of resources as much as possible without losing vital information.

In the specific case of images in a responsive SAP Commerce accelerators storefront, breakpoints are interpreted via `image populators`. Whereas a standard `imagecomponent` element simply holds and displays a single image, responsive components such as the `SimpleResponsiveBanner component` (which is a concrete implementation of the `AbstractResponsiveBannerComponent abstract class`) map individual image files to their respective breakpoints. These mappings are stored in the `Imager.min.js` script library—and they are not necessarily one-to-one; a tablet-sized image may or may not also work for desktop or for mobile, depending on the design.
This more conditional, keypair-based mapping is where image populators come in. The SAP Commerce Accelerator provides two, `ResponsiveMediaContainerPopulator` and `ResponsiveImagePopulator`. The former sends image-size parameters, and the latter populates the appropriate components with the appropriate images.

`ResponsiveMediaContainerPopulator` then passes the sorted results to `Imager.min.js`, which looks for any and all `<img>` elements that have the `js-responsive-image` class attribute in the web browser’s Document Object Model (DOM). When it finds one, it renders images using the parameters in `Imager.min.js`, such as those shown in Listing 5.1.

```
<img class="js-responsive-image" data-media="{"100":"Image of width 100x50",
"200":"Image of width 200x50",
"400":"Image of width 400x50",
"600":"Image of width 600x50",
"800":"Image of width 800x50",
"1000":"Image of width 1000x50"}
alt="">
```

Listing 5.1 Responsive Image Sizing Parameters in `Imager.min.js`

By default, the system maps and populates images by means of naming conventions, with filenames ending with the size of the images: _800x300, for example. `Imager.min.js` checks for these suffixes first. The system also offers a fallback mechanism, leveraging the `mediaFormat` qualifiers defined in the `spring.xml` file of the `acceleratorfacades` extension. Here the breakpoint is based on width because, as you can see in Listing 5.1, `Imager.min.js` uses image width to map images to breakpoints. The named breakpoint widths appear as shown in Listing 5.2 in `spring.xml`.

```
<alias alias="responsiveImageFormats" name="responsiveImageFormats" />
<util:map id="responsiveImageFormats" map-class="java.util.LinkedHashMap" value-type="java.lang.Integer">
<entry key="widescreen" value="1200"/>
<entry key="desktop" value="992"/>
<entry key="tablet" value="768"/>
<entry key="mobile" value="480"/>
</util:map>
```

Listing 5.2 Named Breakpoints for Responsive Pages

An ImpEx job for `SimpleResponsiveBanner` and its associated types and media items, then, would appear as shown in Listing 5.3.

```
INSERT_UPDATE Media;mediaFormat(qualifier);code[unique=true];@media[translator=de.hybris.platform.impex.jalo.media.MediaDataTranslator][forceWrite=true];realfilename;altText;mime[default='image/jpeg'];@contentCV[unique=true];folder(qualifier)[default=images];mobile;Elec_480x320_HomeSpeed_EN_01_480W.jpg;$siteResource/images/banners/homepage/responsive/Elec_480x320_HomeSpeed_EN_01_480W.jpg;Elec_480x320_HomeSpeed_EN_01_480W.jpg;"Fast Precise";
tablet;Elec_770x350_HomeSpeed_EN_01_770W.jpg;$siteResource/images/banners/homepage/responsive/Elec_770x350_HomeSpeed_EN_01_770W.jpg;Elec_770x350_HomeSpeed_EN_01_770W.jpg;"Fast Precise";
desktop;Elec_960x330_HomeSpeed_EN_01_960W.jpg;$siteResource/images/banners/homepage/responsive/Elec_960x330_HomeSpeed_EN_01_960W.jpg;Elec_960x330_HomeSpeed_EN_01_960W.jpg;"Fast Precise";
widescreen;Elec_1400x440_HomeSpeed_EN_01_1400W.jpg;$siteResource/images/banners/homepage/responsive/Elec_1400x440_HomeSpeed_EN_01_1400W.jpg;Elec_1400x440_HomeSpeed_EN_01_1400W.jpg;"Fast Precise";
```

Listing 5.3 ImpEx Job for Responsive Banner

Finally, it bears mentioning that SAP Commerce accelerator storefronts allow for themes, which allow the user to set global colors and typography for their storefronts in the variables.less (for global settings) and theme-variables.less (for theme-specific settings) files. For example, the `@link-color` variable sets the color of links, and the `@text-color` variable sets the default color of text. You can also set variables for brand-specific colors (primary, secondary, and tertiary) within a theme to create brand-specific experiences within a shared responsive storefront.
With the key points of responsive web layouts now clear, and before we move on to SAP Commerce’s tools for managing web content, let’s take a closer look at the structure of an SAP Commerce accelerator storefront web application.

5.2 Web Content Management System

SAP Commerce uses the terms content management system and web content management system somewhat interchangeably, both because much if not all of the content you are managing will be deployed to the web in some manner or another and because the management tools themselves are web-based. In this section, we’ll discuss these tools, as well as the overall content strategies and structures that they enable.

There are two interfaces for web content management in SAP Commerce—the CMS Cockpit and SmartEdit—as well as some relevant functionality in the administration perspective in the backoffice (as shown in Figure 5.4).

Business users creating and editing pages from day to day will do so primarily in SmartEdit, which we will cover in the next section. Here we will focus on the other two interfaces, but before we can do that, we’ll discuss the overall structure of web content in an SAP Commerce accelerator storefront. After all, we can’t talk about managing content until we understand what we’re managing.

Individual elements in an SAP Commerce accelerator storefront—banners, product information, forms, search results, buttons, and so on—are components, defined in their purposes and behaviors by different component types. Components are modeled as individual, modular, reusable parts that can be managed centrally and used repeatedly throughout the customer experience. If your footer links and terms and conditions, for example, are modeled as components, then you can easily include them in your promotional communications or in new pages you create; just as importantly, you can update them a single time in your content management system and see the changes reflected in every location where the content appears.

Components are flexible in terms of the amount of information, visual or otherwise, that they communicate; as such, customers do not necessarily know where one component ends and another begins. In the electronics store, for example, the image carousel at the top of the page houses multiple images within a single component, whereas the content blocks lower on the page are multiple components, each holding a single image, as shown in Figure 5.1 earlier.

But this component-based approach does not mean simply transcluding your header and footer content on each page. It also means reusing banners, product imagery, and other assets on, for example, your homepage, your product listing page, your product detail page, and your promotional pages.

When developing new elements or pages for your site, you want to make sure, whenever possible, to componentize them; that is, you want as many elements as possible in your storefront to be modular components rather than static, hard-coded web content. This is helpful if you want to manage these elements centrally and deploy them to multiple areas of your storefront, and it is necessary if you want to take full advantage of personalization: swapping out some specific asset for some other asset for a specific group of customers or when a customer adds a certain product to his cart.

In practice, this principle tends to go by the wayside when project timelines get tight or other areas of the project require more attention than you’d originally planned. Although not ideal, this compromise can work so long as you are intentional about which parts of your customer experience need to be componentized and which do not. Perhaps your header and footer will be fairly static and therefore can be hard-coded. Perhaps your checkout experience is specific enough to your line (or lines) of business that you will not be changing or personalizing it. In such cases, leaving some
components static may be perfectly acceptable, at least during your first phase of deployment. If too much (or all) of a given page becomes too static, however, you will be missing out on many of benefits of SAP Commerce’s personalization features, which we will discuss in Section 5.4.

Components are vital to the structure of your storefront, but they are not the navigational unit that customers most readily think of when they interact with your website. Rather, most customers imagine your site’s structure in terms of pages, the effectual combination of structured content (componentized or not) that appears when a customer arrives at a certain URL. To understand pages, we will first need to understand page templates and page types, which are themselves divided into five important page types:

1. **Content pages**
   These are built primarily by hand by content managers (though they can be personalized for specific users, as we will discuss later).

2. **Category pages**
   These list products for a given category or a given set of search parameters and are therefore built partially by Solr search (which we will discuss more thoroughly in Chapter 8).

3. **Product pages**
   These show individual products or bundles and are therefore built partially by your product content catalog(s).

4. **Abstract pages**
   These hold functionality such as checkout, managing saved carts, creating returns, initiating password resets, and creating support tickets.

5. **Document pages**
   These generate emails and forms to keep them in line with the look, feel, and functionality of the rest of your storefront.

It is important to emphasize that you have the highest degree of direct, day-to-day control over content pages, such as the homepage. You can add personalized or promotional content to checkout pages, product detail pages, product listing pages, and so on—but these have certain elements that need to be structured and built by the system, whether that means ordered search results in the case of a product listing page (as discussed in Chapter 8) or address and payment forms in the case of a checkout page (as discussed in Chapter 3). Content pages do not necessarily have any of these more system-assembled or instrumental elements, and content pages are therefore the best place to start when you begin to create new pages in the content management system.

The SAP Commerce accelerator storefronts come bundled with some page templates for each of the page types: landing page templates for homepages, product detail pages for product listings, and even prebuilt layouts for frequently asked questions and account management pages. Page templates include JSP-based layouts, sizing information for each slot into which components can be placed, and rules for which component type can be placed into which slot.

This can be as general as saying that a given component slot must contain an image or as specific as saying that a given component slot must contain an image of a type that specifies size, format, and origin. It’s up to you to curate your media types, but your inclusion and exclusion rules can be as intentional and proscriptive as you like, so long as your media items are organized and tagged accordingly. Being prescriptive about your template and slot rules can prevent content managers from inadvertently breaking the page but also can limit their ability to be inventive within the rules you’ve set.

Your goal always should be to balance these two factors because without explicit exclusion or inclusion rules, the system will always tend toward adaptability. If you have a slot for a full-width image, for example, and you do not specify some media type that contains images of the correct dimensions, then a content manager will be able to insert any image to which they have access into that slot, which would result in a stretched, blurred, or otherwise suboptimal asset.

This same flexibility means that an attentive and inventive content manager can achieve a large number of discrete looks and feels from a relatively small number of page templates. Removing that full-width banner entirely will not result in a large field of blank white space but rather in the components below that one rising to meet the header, thereby creating an effectually different page layout, as shown in Figure 5.5.

Page templates govern much of the layouts that customers see when they browse your storefront, but again, it is the pages themselves that customers will interact with, moment to moment, and that content managers will build during and after development. An active, potentially browsable instance of a given page template, with one or multiple URLs to reach it, is a page. Where a page template says an image goes here, a page says this image goes here. Landing page 2 is a page template; the homepage is a page.
That’s in the case of content pages. In the case of product pages, even the page says the product content goes here rather than this product’s content goes here. There is generally only a single product page on a given SAP Commerce accelerator storefront, and, as described earlier, the specific product information being displayed will originate in the product catalog rather than the content catalog.

With all of that said, we can move on to the admin perspective in the backoffice and the Web Content Management System (WCMS) Cockpit, beginning with the former—a view for your content catalogs, data model, and overall structure. You can see this data set under the WCMS node (see Figure 5.6) when logged in as an administrator or as any other backoffice user with access to content catalogs and their associated types.

Some important functions that are not available in SmartEdit are available here, as follows:

- Creating content catalogs (and catalog versions)
- Creating homepages
- Restoring deleted pages
- Permanently deleting pages “deleted” in SmartEdit

It is also generally here that more senior or more technical content managers can edit the details of catalogs, catalog versions, and page templates, as well as view restrictions more holistically. (We’ll discuss restrictions in Section 5.3.4, as well as in Chapter 9).

You may have noticed that this view does not cover intuitive, what-you-see-is-what-you-get (WYSIWYG) page editing. That functionality is covered by SmartEdit—as well as by the CMS Cockpit, the original purpose-built interface for content catalogs in SAP Commerce. The CMS Cockpit has three important perspectives: WCMS page view, live edit, and navigation. In the WCMS page view, shown in Figure 5.7, content managers can edit pages in a wireframe-like view.

Live edit is essentially an earlier iteration of SmartEdit, allowing users to preview pages (with personalization rules if desired) and edit those pages in place. In a system
that has SmartEdit running, there is no reason to prefer live edit: though SmartEdit has not yet achieved full feature parity with the old cockpit, the features that it lacks are covered by the admin perspective in the backoffice, as described previously.

Figure 5.7 WCMS Page View in WCMS Cockpit

Navigation, meanwhile, allows content managers to create and edit the navigation nodes that make up the headers and footers of pages. Although the header nodes typically mirror top-level product categories and/or page hierarchies, it’s important to keep in mind that they are in fact entirely independent of these. They are purely content catalog items, even if they link to items in the product catalog.

Because navigation nodes also can be created and modified in SmartEdit, let’s now move on to what is, for most users, the primary content manager interface in SAP Commerce. Let’s talk about SmartEdit, both in terms of day-to-day use and in terms of configuration and customization.

5.3 SmartEdit

SmartEdit allows content managers to create new pages from existing templates, add, edit, and remove those components via drag and drop, and test personalized experiences within those pages, all from a single scalable user interface. In this section, we’ll discuss the technology behind SmartEdit, its core features and functions, the steps necessary to customize the tool, and, perhaps most importantly, the groundwork that SmartEdit lays for personalized customer experiences.

From a technical perspective, SmartEdit is a UI for managing web pages and a JavaScript framework that allows developers to connect pages to that UI, following a model-view-controller (MVC) approach and using AngularJS modules to compose HTML, CSS, and JavaScript fragments.

This UI and this framework are both natively and automatically connected to SAP Commerce accelerator storefront pages, but in principle they can be connected to other web frontends as well.

5.3.1 Framework and Architecture

Before we discuss SmartEdit in greater technical detail, let’s take a moment to discuss Grunt, third-party software used to build SmartEdit extensions. Grunt organizes and executes plugins, configuring, ordering, and grouping them via the settings in Gruntfile.js, located in the root folder of all SmartEdit extensions that make use of JavaScript.

Grunt plugins format JavaScript and HTML files (often adding semicolons, which are easy to accidentally exclude), analyze JavaScript syntax, translate HTML templates to JavaScript modules, minify AngularJS JavaScript files, and manage source concentration, unit and integration testing, and generation of CSS files via LESS.

The primary function of Grunt in a SmartEdit context is for building extensions, but you can also use plugins that watch for changes in the running system (the watcher plugin) and integrate your system with other systems for debugging and testing (the connect plugin). To add these plugins to your Gruntfile.js, add the code in Listing 5.4.

watch: {
  xyz: {
    files: ['Gruntfile.js', 'web/features/**/*', 'jsTests/**/*'],
    tasks: ['dev'],
    options: {
      livereload: true,
      server: { base: '.' } // optional
    }
  }
}
You call Grunt using the nomancillary extension of the npm-ancillary module. You'll need to include this extension in the extensioninfo.xml file for each relevant Java-Script extension. The extensioninfo.xml file in the cmssmartedit extension, for example, contains the parameters from Listing 5.5.

```xml
<extensioninfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="extensioninfo.xsd">
  <extension name="cmssmartedit"
    abstractclassprefix="Generated"
    classprefix="CmsSmartedit"
    managername="CmsSmarteditManager"
    managersuperclass="de.hybris.platform.jalo.extension.Extension"
  >
    <requires-extension name="npmancillary" />
    <requires-extension name="smartedit" />
    <coremodule packageroot="de.hybris.cmssmartedit"
      manager="de.hybris.cmssmartedit.jalo.CmsSmarteditManager"
      generated="true"
    />
    <webmodule jspcompile="false" webroot="/cmssmartedit"/>
  </extension>
</extensioninfo>
```

Listing 5.5 Relevant Additions to extensioninfo.xml

This connects your extension to the extensioninfo.xml file, which in turn contains and calls the gruntBuild macro. This macro creates a symbolic link (called node_modules) between the nomancillary extension and your own, as shown in Listing 5.6.

```xml
<?xml version="1.0"?>
<project name="cmssmartedit_buildcallbacks">
  <macrodef name="cmssmartedit_before_build">
    <sequential>
      <linkToSELibraries path="${ext.cmssmartedit.path}"/>
    </sequential>
  </macrodef>
</project>
```

Listing 5.6 gruntBuild macro in extensioninfo.xml

You can add additional scripts and configurations to your Gruntfile.js file as needed, as shown in Listing 5.4.
Listing 5.6 Making Relevant Additions to cmssmartedit

With an understanding of Grunt, we can now move on to the data model and configuration options of SmartEdit itself.

SmartEdit modules contain three key types of features, as shown in Figure 5.8:
1. Decorators, which add rendering and/or behavior to components. (In the code, these behavior triggers are sometimes referred to as directives.)
2. Contextual menu items, which are added to the core contextual menu decorator, with each item performing a different action.
3. Toolbar actions, which enact behavior when triggered by contextual menu items.

To add a new decorator, start by creating an AngularJS module. In this example, let’s name it SampleDecorator; call the decorator (or directive) itself sample. (Note that the name of your module must end in the Decorator suffix.) Save the new module in the web/features/sampleDecorator directory. Our goal will be to create a new decorator activated by the customer’s mouse moving over the attached SmartEdit element. In the code, these steps would appear as shown in Listing 5.7.

```javascript
angular.module('SampleDecorator', ['myModuleTemplates'])
  .directive('sample', ['$timeout', function($timeout) {
    return {
      templateUrl: 'web/features/sampleDecorator/sampleDecoratorTemplate.html',
      restrict: 'C',
      transclude: true,
      replace: false,
      scope: {
        smarteditComponentId: '@',
        smarteditComponentType: '@',
        active: '='
      },
      link: function($scope, element, attrs) {
        $scope.visible = false;
        $scope.mouseleave = function() {
          $timeout(function() {
            $scope.visible = false;
          }, 1000);
        };
        $scope.mouseenter = function() {
          $scope.visible = true;
        };
      }
    };
  }]);
```

Listing 5.7 AngularJS Module for New Decorator

Let’s break down the results of that snippet. First, the code sets the value for templateUrl so that the compiled directive can locate the decorator template, as shown in Listing 5.8.
angular.module('sampleDecorator', function($timeout) {
  templateUrl: 'web/features/sampleDecorator/sampleDecoratorTemplate.html',
}
Listing 5.8 Setting Value for templateUrl

The code also sets the value for the restrict property to 'C', which will cause the directive to invoke only the CSS class type:

restrict: 'C',

Further, the code sets the transclude property to true and the replace property to false. This means that the decorator will append the existing DOM rather than replace it. In other words, this decorator will not override the entire UI but will simply add to it:

transclude: true,
replace: false,

The code in Listing 5.7 also adds the following three parameters, which we show again in Listing 5.9:
1. smarteditComponentType, which defines the type of component being extracted from the DOM
2. smarteditComponentId, which defines the ID of the specific smarteditComponent being extracted
3. active, which acts as a toggle to activate or deactivate the decorator

scope: {
  smarteditComponentId: '@',
  smarteditComponentType: '@',
  active: '='
},

Listing 5.9 Adding Parameters and Enabling Transclude

Finally, the code sets the link function, which attaches event listeners to DOM elements, watches model properties for changes, and updates the DOM. When the directive is called, it acts as a controller. Listing 5.10 sets three values for the link function:

$scope.visible = false;

$timeout(function() {
  $scope.visible = false;
}, 1000);

$scope.mouseleave = function() {
  $scope.visible = false;
};

$scope.mouseenter = function() {
  $scope.visible = true;
}

Listing 5.10 Adding Link Function and Its Associated Values 

Note that any module you create for a decorator will need to contain the <div data-ng-transclude> tag, which tells the directive where to inject its appended—or, more exactly, transcluded—content. (We say that this content is transcluded because it exists in a content repository and can be called in multiple contexts, and because it is one part among several in a componentized presentation layer.)

Now, with all of this done, we can create a template. Call it sampledecoratorTemplate.html (decorator template names should always end in the suffix Template.html, for a reason that will be clear in a moment) and place it under web/features/sampleDecorator/SampleDecoratorTemplate.html. This template will define mouseenter and mouseleave event handlers to properly control the visibility of the decorator, which would appear in the code as shown in Listing 5.11.

1. visible is set to false, meaning that the decorator is not initially active.
2. When the customer’s mouse hovers over (mouseenter), the decorator becomes visible.
3. When the customer’s mouse is no longer hovering over (mouseleave), the decorator is no longer visible.

link: function($scope, element, attrs) {
  $scope.visible = false;
  $scope.mouseleave = function() {
    $timeout(function() {
      $scope.visible = false;
    }, 1000);
  };
  $scope.mouseenter = function() {
    $scope.visible = true;
  };
}
With this done, next write the decorator to the module you created in the `web/features` folder. Create a module called `myModule` and add the decorator as a dependency—while also adding `decoratorServiceModule`, which will allow you to wire the decorator to a component type in the next step, as shown in Listing 5.12.

```javascript
angular.module('myModule', ['sampleDecorator', 'decoratorServiceModule'])
```

**Listing 5.12 Writing New Decorator to Module**

You can now use the `addMappings` method of `decoratorService` to wire the decorator to one or more components, as shown in Listing 5.13.

```javascript
.run(function(decoratorService) {
  decoratorService.addMappings({
    'sampleComponent': ['sample']
  });
});
```

**Listing 5.13 Wiring Decorator to Component**

If you want to wire decorators to multiple components, you can do so by listing the exact decorator names:

```javascript
'componentType1': ['decorator1', 'decorator2']
```

You can also specify a component type and apply the decorator to all components of that type:

```javascript
'.*Type1': ['decorator1']
```

Next, let’s use `Gruntfile.js` to create an AngularJS module for the decorator templates, converting the HTML JavaScript modules as shown in Listing 5.14.

```javascript
ngtemplates: {
  options: {
    standalone: true,
    module: 'myModuleTemplates'
  },
  run: {
    src: ['web/features/**/*Template.html'],
    dest: 'jsTarget/web/features/templates.js'
  }
}
```

**Listing 5.14 Creating AngularJS Module for New Decorator Template(s)**

On execution of this task, the system will find all templates with names that end in `Templates.html` (which is why you used that suffix on your own new template) and load them into the `templates.js` file. The templates will reside in an AngularJS module called `myModuleTemplates`. You can then wire your decorator modules to the `myModule.js` module, thereby integrating your templates into the SmartEdit experience and eliminating the need for real-time AJAX calls.

Note that this step is only necessary if you have created your own new module. If you have an extension generated from the out-of-the-box `ysmarteditmodule` template, then this wiring will be complete as soon as you have activated Grunt. The same caveat applies to the next step, in which we will (at last) expose the module to the SmartEdit application.

If you are dealing with a newly added module, then you will need to create a unified JavaScript file to expose it. This file will live in the `web/webroot/js` directory. We will concatenate a file using the Grunt build plugin and save it in `myModule.js` by adding the contents of Listing 5.15 to `Gruntfile.js`.

Or you can wire the decorator to all components, but with exceptions (the exception here being any component with “Type1” in its name):

```javascript
'^((?!Type1).)*$': ['decorator1']
```

**Listing 5.15 Wiring Decorator to Component with Exceptions**
5.3.2 Content Management

SmartEdit offers four main perspectives for content managers:

1. Preview, which allows the user to view pages as they would appear to a customer without any personalization
2. Basic edit, which allows for quick, drag-and-drop changes to the positions and properties of simple components such as banners
3. Advanced edit, which allows for the editing and movement of all components, including navigation
4. Personalization, which allows the user to view the page as a customer would, with personalization rules in place

You can and should restrict these different perspectives to different content managers (and in the case of preview and personalization, perhaps testers or business stakeholders, who are not interested in making changes but who would like to see new or updated content before it goes live). By doing so, you can offer basic editing to those who are building simple pages, advanced editing to those who are making more complicated, more custom, or more conditional changes, and so on.

While building and testing your storefront, you should be careful to properly componentize your content, as discussed earlier in this chapter. None of the page elements or other components that you have hard-coded will be editable via SmartEdit or personalizable via restrictions.

5.3.3 Components

Accelerator storefronts ship with components, pages, and page templates prebuilt. As discussed earlier in this chapter, these components can be added, edited, and rearranged using SmartEdit. Developers will frequently create their own new component types, sometimes using the SAP Commerce accelerators components as bases and sometimes starting from scratch. These new components will need to be enabled for SmartEdit, as described earlier in this chapter.

The component types enabled for SmartEdit by default, as well as their functions, are outlined in Table 5.1.

A common symptom of insufficiently componentized storefronts is the overuse of paragraph components. These are intended for rich text (perhaps with tables, hyperlinks, or embedded images) and can be edited via a WYSIWYG interface in SmartEdit—but because they can contain rich formatting such as tables and hyperlinks and because they can be edited directly in HTML format as well as via WYSIWYG, paragraph components can also include more complicated HTML or scripts. Some developers and content managers will therefore insert custom “components” as large blocks of arbitrary HTML in paragraph components.

This is not a best practice, for three important reasons. First and foremost, this practice is intensely sensitive to keystroke error, which could result in broken components, poorly optimized scripts, or soft security practices. Second, should you notice these issues, it can be very difficult to track down the offending code or even to deactivate only the part that is malfunctioning without also deactivating some other, perhaps critical, piece of functionality or content. Third, and just as importantly, overusing paragraph components in this way denies the business the opportunity to define, design, and refine the component types that they should have. If there is some storefront element that is critical to your business but that is not represented or accounted for in the accelerator storefronts, then your project should include building it. The results will be better, you will have more management and personalization options, and you will save yourself a good deal of stress during testing and deployment.

SmartEdit is at its most useful when content managers are empowered to build new pages and variations on existing ones without having to involve developers or internal IT—and the interface allows this, so long as your design and implementation does as well.

Listing 5.15 Concatenating New File Using Grunt Build Plugin
5 Customer Experience Management

<table>
<thead>
<tr>
<th>Component Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account navigation</td>
<td>Provides the customer with access to self-service functionality, as described in Chapter 7.</td>
</tr>
<tr>
<td>Category navigation</td>
<td>Displays the product categories and/or other items that you have added to your core storefront navigation, as described in Section 5.3.5.</td>
</tr>
<tr>
<td>Footer navigation</td>
<td>Displays the footer content, usually including static links to contact information, frequently asked questions, and so on. This content is considered part of navigation and is covered in Section 5.3.5.</td>
</tr>
<tr>
<td>Image map</td>
<td>An image with a link embedded in some or all of its pixels, configurable via HTML.</td>
</tr>
<tr>
<td>Link</td>
<td>The component is nested under navigation nodes to provide connections to other parts of the storefront, as described in Section 5.3.5.</td>
</tr>
<tr>
<td>Mini cart</td>
<td>Summarizes the current contents and total value of the customer’s cart.</td>
</tr>
<tr>
<td>Paragraph</td>
<td>A flexible component, editable via SmartEdit or any other content management interface in SAP Commerce, that can contain plain text, rich text, tables, images, hyperlinks, and so on.</td>
</tr>
<tr>
<td>Product carousel</td>
<td>Displays a set of products, with the attributes of those products (such as name, price, and image) being drawn directly from the product catalog.</td>
</tr>
<tr>
<td>Product reference</td>
<td>Shows products related to one another, in manners such as cross-selling, upselling, and accessories, as described in Chapter 3 and Chapter 4.</td>
</tr>
<tr>
<td>Search box</td>
<td>The component that allows customers to search your indexed search data for keywords.</td>
</tr>
<tr>
<td>Simple banner</td>
<td>Shows an image and can link to a URL when clicked. These components also contain metadata and, like all media in the SAP Commerce system, can be protected using security rules.</td>
</tr>
<tr>
<td>Simple responsive banner</td>
<td>Serves the same function as a simple banner but includes image files and parameters for each of your responsive breakpoints.</td>
</tr>
</tbody>
</table>

Table 5.1 Component Types

5.3.4 Restrictions Management

Restrictions mark a piece of content as being conditional—that is, being shown or not shown based on some specific parameter. "Time restrictions," for example, mean that the component will only appear within a specific time range. "Category restrictions" mean that the component will show only if the user is currently viewing a certain category, thereby allowing bespoke content to be added to otherwise system-built pages, such as product listing pages.

In SmartEdit, users can add restrictions to components by selecting the appropriate component and then clicking the button, as shown in Figure 5.9. (For a more detailed understanding of restrictions, see Chapter 9.)

Figure 5.9 Adding Restriction in SmartEdit

One specific restriction type that bears mentioning in this context is the CMS action restriction, which, as its name implies, adds a CMS action to the component. This means adding some behavior to a component, over and above linking to some other part of the storefront via a click. The most commonly used CMS actions are as follows:

- **PickUpInStoreActionModel**, which initiates a buy online, pickup in store checkout process (as described in Chapter 3)
ShareOnSocialNetworkActionModel, which opens an interface for sharing a product or page on one or more social networks.

AddToCartActionModel, which instantly adds a product to the customer’s cart.

SimpleCMSActionModel acts as the base for all CMS actions, and allows you to create your own new ones. You would begin doing this by defining a new item type, which we’ll call MyNewAction, as shown in Listing 5.16.

```xml
<itemtype code="MyNewAction" jaloclass="de.hybris.platform.acceleratorcms.jalo.actions.MyNewAction" extends="SimpleCMSAction" autocreate="true" generate="true">
</itemtype>
```

**Listing 5.16** Defining MyNewAction

To test this, wire it a simple visual element, a button that says Click Me! To do so, add the following to your storefront extension:

```xml
<%@ page trimDirectiveWhitespaces="true" %>
<button type="button">Click Me!</button>
```

You can, of course, wire your new action to a more sophisticated component than that. Listing 5.17, for example, shows how you would connect it to ProductAddToCartComponent in the accelerator storefronts via ImpEx.

```xml
# this relation is used by CMSActionRestriction, by adding ActionRestriction to your action
INSERT_UPDATE ApplicableCmsActionsTypeForCmsComponent;target(code)[unique=true];source(code)[unique=true];ProductAddToCartComponent;MyNewAction;

# CMS ACTION Restrictions
INSERT_UPDATE CMSActionRestriction;$contentCV[unique=true];uid[unique=true];name;ActionRestriction;A CMSAction restriction;

# CMS ProductAddToCart Components
INSERT_UPDATE ProductAddToCartComponent;$contentCV[unique=true];uid[unique=true];name;actions($actionRef);&componentRef;AddToCart;Product Add To Cart;PickUpInStoreAction,AddToCartAction,ShareOnSocialNetworkAction,MyNewAction;AddToCart
```

**Listing 5.17** Connecting New Action to ProductAddToCartComponent

Finally, you need to add your new action to the appropriate component view, using the syntax shown in Listing 5.18 (which comes from the example in the ProductAddToCartPanel.tag SAP Commerce accelerators file).

```xml
...<%@ taglib prefix="action" tagdir="/WEB-INF/tags/desktop/action" %>
...
<div id="actions-container-for-#{component.uid}" class="productAddToCartPanelContainer clearfix">
<ul class="productAddToCartPanel clearfix">
<action:actions element="li" styleClass="productAddToCartPanelItem span-5" parentComponent="#{component.uid}"/>
</ul>
</div>
```

**Listing 5.18** Sample Syntax from ProductAddToCartPanel.tag

### 5.3.5 Navigation

Navigation components are unique insofar as they are specific to menus, usually shared across pages, that allow the customer to move around the site. Although banners and other components will frequently include links to other pages, navigation components exist primarily for linking; as such, they should be designed with some intention. We generally recommend linking to your most important product categories and to any important nonproduct pages that you might have, from learning resources to (especially in the case of highly regulated industries) policy information. Product and product-related links typically go in your header, whereas Contact Us, Frequently Asked Questions, and copyright boilerplate typically go in your footer.

Navigation components can be edited only from the advanced edit perspective, as shown in Figure 5.10.
In addition, the onsite search bar is itself a configurable component. Employees in the CMS Cockpit or SmartEdit can determine how long the search engine will wait for before making suggestions, as well as the kinds of suggestions it will then offer—suggesting keywords or specific products or both if autocomplete is turned on and, after the search is executed, following up with “Did you mean?” functionality if spell check is turned on. For a more thorough discussion of these features, as well as the search functionality attached to them, see Chapter 8.

There are some settings that are unique to the search bar component, however. Product recommendations are handled at the SAP Commerce level rather than the search engine level, so they can be enabled only from the content management side; likewise, the settings for delaying before returning results can only be set here. The system allows you to set the number of characters that the search bar will wait for (three by default) and the number of milliseconds (500 by default) that the storefront will then wait before returning results.

You might be wondering why anyone would want to intentionally delay autocomplete results. After all, reducing your response time for search can be key to providing a good customer experience, and long load times tend to greatly increase bounce rates. But, perhaps counterintuitively, these delay settings can help to improve performance and increase customer engagement.

This is because returning autocomplete results as quickly as possible, after every single keystroke, generally results in verbose, unwieldy suggestions, while also requiring the storefront to query the search engine after every single keystroke. It’s generally better, then, to wait for a better idea of what the customer is looking for and also to wait for some indication that they are no longer typing. This feature allows you to account for both of those factors and to fine-tune the delay time based on your search engine’s base response time and your customers’ specific expectations. (For more on both these topics, once again, see Chapter 8).

5.3.6 User Accounts and User Groups

User groups are just what they sound like: groups of users organized in the SAP Commerce system. As we discuss in Chapter 9, user groups can contain employees or customers—but in the context of SmartEdit, we’re concerned with customer-facing user groups. By adding user group restrictions to specific pieces of content, from a single banner to an entire page or section of your storefront, you can provide customers with a highly personal experience even if you have yet to set up automation.

In the personalization perspective, users can choose to view a page in the context of one or more user groups, each representing a category of customers to which you want to tailor your storefront experience: customers who are loyal to some specific brand, big spenders, deal hunters, or any other group that, in your view, drives (or should drive) customer behavior.
Most importantly, this view allows you to see what the page would look like to a customer who is a member of multiple user groups. As shown in Figure 5.11, the content that appears as a result of membership in each applicable user group will be color-coded, providing users with a transparent view of what customers will experience, why, and where the content might potentially clash.

Figure 5.11 Personalization in SmartEdit, Showing Components for Two Customer Segments

5.4 Personalization

A storefront, however well-designed, only goes so far if the content is not relevant to the customers using it—and because customers are not identical, the content that you serve to them shouldn’t be, either. The personalization module helps you to tailor your experience to customers’ preferences, expectations, and needs quickly and effectively. We’ll start by going through the overall architecture of personalization in SAP Commerce before going into more detail on both basic and advanced configuration of those features.

5.4.1 Personalization Architecture

Personalization relies primarily on two extensions, personalizationservices and personalizationfacades, with the former handling core functionality and logic and the latter exposing those functionalities by providing appropriate facades and exposing POJOs for integration purposes.

The key objects used by the personalizationservices extension are as follows:

- CxSegments, which acts as the main object for customer classification, bridging the gap between customer-facing user groups (as discussed above, as well as in Chapter 9) with SmartEdit-specific segments
- CxAbstractAction, which is the main class for executing behavior, but which does not itself define that behavior, relying instead on other classes to extend it
- CxCustomization, which groups restricted components for the appropriate segment, such as SummerSale2019
- CxVariation, which defines variations on the customizations defined in CxCustomization—e.g., SummerSale2019MensWear
- CxVariation, which stores the segments assigned to a user so that the system will remember which customers belong to which segment

The personalizationfacades extension then converts segments to storefront content, using the following converter and populator types, as defined in the personalizationfacades-beans.xml file:

- populator, which defines the source values the target object
- populatorList, which groups the relevant populators
- configurableConverter, which uses the populatorList to define the populators that will be invoked
- configurableSubtypeConverter and configurableSubtypeAwareConverter, which extend configurableConverter, defining the parent type of the object(s) to be converted and sorting those objects as actions or triggers

The relations among these objects, then, are as shown in Figure 5.12.
5.4.2 Configuring Personalization

As alluded to earlier, triggers activate variations for specific user groups. CxTriggerService executes the CxTriggerStrategy element registered in the system. CxDefaultTriggerStrategy uses CxDefaultTrigger to deploy a default variation—in other words, your storefront without personalizations—and DefaultSegmentTriggerStrategy then maps other variations to the appropriate customer segments. DefaultSegmentTriggerStrategy also sets a threshold for customer affinity so that only truly appropriate segments will be considered. You can set this threshold using the personalizationservices.segment.trigger.strategy.min.affinity property.

Finally, ExpressionTriggerStrategy handles cases wherein multiple segments need to be combined and defines the logic for grouping operators (and as opposed to or), element negation, and segment definition. A JSON expression of this logic might look like Listing 5.19.

```
"expression": {
  "type": "groupExpression",
  "elements": [
    {
      "type": "segmentExpression",
      "code": "Men"
    },
    {
      "type": "groupExpression",
      "elements": [
        {
          "type": "segmentExpression",
          "code": "Women"
        },
        {
          "type": "negationExpression",
          "element": {
            "type": "segmentExpression",
            "code": "HatsFan"
          }
        }
      ],
      "operator": "AND"
    },
    {
      "type": "negationExpression",
      "element": {
        "type": "segmentExpression",
        "code": "HatsFan"
      }
    }
  ],
  "operator": "OR"
}
```

Listing 5.19 Logic for Combining Multiple Segments, as Expressed in JSON

5.4.3 Extending Personalization

If you want to extend the logic of personalization converters, you can do so in the beans for configurableSubtypeConverter and configurableSubtypeAwareConverter, as shown in Listing 5.20.

```
<bean id="defaultCxTriggerConfigurableConverter" parent="cxSubtypeAwareConverter">
  <property name="defaultOptions" value=""de.hybris.platform.personalizationfacades.enums.TriggerConversionOptions">BASE</value>
  <property name="markerClass" value=""de.hybris.platform.personalizationfacades.converteroptions.TriggerConversionOptions">BASE</property>
</bean>
```
Listing 5.20 Extending Personalization Converter Logic

Using these beans, you can define one or more of your own `configurableSubtypeConverter` elements, as shown in Listing 5.21. (Be sure to use the same marker class here and in `configurableSubtypeAwareConverter`)

```xml
<bean id="defaultCxSegmentTriggerConfigurableConverter" parent="cxSubtypeConverter">
    <constructor-arg name="markerClass" value="de.hybris.platform.personalizationfacades.trigger.converters.TriggerConverter" />
    <constructor-arg name="sourceClass" value="de.hybris.platform.personalizationservices.model.CxSegmentTriggerModel" />
    <property name="targetClass" value="de.hybris.platform.personalizationfacades.data.SegmentTriggerData" />
    <property name="populators">
        <map key-type="de.hybris.platform.personalizationfacades.enums.TriggerConversionOptions">
            <entry key="BASE" value-ref="cxTriggerPopulators" />
            <entry key="FOR_SEGMENT" value-ref="cxSegmentTriggerForSegmentPopulators" />
            <entry key="FOR_VARIATION" value-ref="cxSegmentTriggerForVariationPopulators" />
            <entry key="FULL" value-ref="cxSegmentTriggerFullPopulators" />
        </map>
    </property>
</bean>
```

Listing 5.21 Defining `configurableSubtypeConverter`

5.4.4 Advanced Personalization Module

The legacy CMS Cockpit includes an advanced personalization module, allowing SAP Commerce to segment customers into behavioral targeting groups (BTGs), which act as customer-facing user groups built on the fly and without necessarily having to persist past a single session. BTGs also can be configured in the administration perspective in the backoffice if you’re running a compatible SAP Commerce version.

The advanced personalization module creates BTGs using rules—in essence, if/then statements that can contain one or multiple conditions and one or multiple results when those conditions are met. These conditions fall into four main categories (shown as they appear in the backoffice interface in Figure 5.13):

1. **Customer rules**
   These check for account-level information, such as the customer’s country, postal code, gender, or current user group membership.

2. **Order rules**
   These check for details in the customer’s order history, such as the products, categories of products, or total value of previous orders.

3. **Cart rules**
   These check the customer’s currently active cart for certain products, categories of products, or quantities of products—or can simply check whether the cart is or is not empty or check its value.

4. **Website rules**
   These check for the customers’ browsing behavior on the SAP Commerce storefront, including what products or categories they have viewed and what pages they have visited—or can check for affiliate links or URL parameters (in the form of keypairs, such as `?emailpromotion=true`).

![Figure 5.13 BTG Rules in Advanced Personalization Module](image)
Note that the first two categories require you to know which customer you’re dealing with because they refer to past orders and customer profile information. Cart rules and website rules, however, do not require a known user. You can extract relevant information about the cart and/or browsing session of an anonymous customer just as effectively as a known one.

It’s also important to note that the values we’re checking for do not need to be absolute. You could, for example, create a customer rule that checks whether the customer’s postal code starts with 606, effectively checking whether they are in the greater Chicago area, rather than zeroing in on one specific zip code.

After evaluating some combination of these conditions, the advanced personalization module then performs one or more actions—adding the customer to one or more user groups and/or showing, hiding, or replacing a page component.

Advanced personalization relies on the btg and btgcockpit extensions, with the former handling the logic of BTGs and the latter governing the UI elements with which employees can manage BTGs and associated rules. BTGs run on a proprietary technology stack, entirely different from the Drools-based rules engine discussed in Chapter 8.

Because the advanced personalization module does not connect to SmartEdit, and because SmartEdit’s personalization features serve a similar function with more flexibility, the older advanced personalization module is considered deprecated functionality. It is also less performant than user-group-based personalization and generally should be used only in earlier versions of SAP Commerce that are incompatible with SmartEdit.

5.5 Context-Driven Services

SAP Commerce Cloud’s context-driven services is a cloud-based personalization and conversion optimization solution for commerce customers. It’s offered as an add-on to SAP Commerce Cloud as well as SAP Commerce on-premise solutions. The context-driven services solution provides real-time customer experience personalization by capturing and analyzing contextual and behavioral data across each customer’s commerce journey. The analyzed data intelligently creates individualized customer experiences, which results in an improved conversion rate, raises order value, and increases customer satisfaction and loyalty.

5.5.1 Foundation

Context-driven services foundation allows businesses to create, maintain, and continually extend customer profiles from a range of data sources. The context-driven services foundation uses short-term and long-term contextual data from SAP Commerce behaviors and interactions to drive the SAP Commerce experience. The current solution captures data from digital channels originating on the web, clickstream data, and commerce data.

It provides attributes, scores, and real-time metrics per customer, from anonymous to fully identified customers. Customer consent is always respected. The context-driven services foundation platform is extensible and leverages a schema-less, flexible data structure to capture new data and insights to enrich and extend customer profiles. The Lambda enricher framework provides full extensibility to utilize information from other SAP data sources such as SAP Customer Profile (formerly, Gigya Profile) and any external third-party systems and channels to create new insights.

Let’s look at the architecture of foundation, different components of foundation, and how it integrates with the rest of context-driven services.
Foundation Architecture
The foundation data journey consists of four main steps: data capture, identity management, context processing, and consumption.

The incoming data is referred to as context and can be either streaming data or transactional data. The clickstream data from web events is mapped to the foundational extensible data model by a profile tagging solution. Transactional data like customer master data, orders, returns, and so on is passed on to SAP Commerce with prebuilt integrations.

Foundation starts to identify the customer from the first interaction. The anonymous user’s context is captured via cookies and known user context via the login information. If the captured context matches an existing profile, it combines the data; otherwise, a new identity is created. Session matching, cross-site matching, and cross-device matching all contribute to building the context, and if the two profiles are identified for the same customer, then they are merged, leveraging the fully automated merge service.

Once the context is built, it’s then processed and analyzed in real time and added as observations, transactions, or insights to the customer profile. Foundation can be extended via additional services to meet specific customer requirements.

The analyzed context can be consumed in two ways:
1. Through the foundation API that allows real-time access to a complete individual customer profile or parts of it
2. Through the foundation’s real-time reporting service that allows you to query the customer data on multiple aggregation levels

Figure 5.14 summarizes the foundation architecture, showcasing the flow from data capture to identity management and context processing to consumption.

Customer profile data such as sessions, interactions, insights, and other relevant data is stored in the document database. The document database has various advantages compared to the traditional relational database, like providing flexibility to extend without redefining the database structure, allowing easy data augmentations to profiles, storing customer data in one place within a single JSON document, and making all customer profile data able to be retrieved with one request.

Figure 5.14  Context-Driven Services Foundation Architecture

Foundation Components
Foundation is built up of multiple components, as follows:

- **Profile tag**
The profile tagging solution is a website tagging solution that is included with context-driven services to effortlessly capture data. It consists of a graphical user interface and JavaScript library.

- **Context service**
This is a big data cluster that stores all raw events in the context service. Each enricher microservice can query the context service to access all the message details (context) for further processing.

- **Enricher microservices**
Enricher microservices process data that flows through a context adapter. Enrichers usually analyze the data then persist insights and observations in the profile document.

- **Data privacy and consent management**
Customer data cannot be collected or used without a customer’s consent. The brand or store owner exposes consent management using context-driven services foundation consent services to customers at any level of data granularity, from global control to fine-grained control of data types and sources.
Consent service
The consent service manages all tenant- and user-specific consents. It creates and returns consent references, which can be used for all requests and operations.

Profile service
Context-driven services foundation provides a dedicated service to manage and retrieve profile documents. Consuming applications use the profile service to retrieve either full profile data or partial profile data.

Real-time reporting service
In parallel to the regular data processing through the context service and enricher microservices, the system sends data to the real-time reporting service. The real-time reporting service of context-driven services foundation allows you to process large event streams (such as clickstreams or commerce data) in real time. Data aggregation happens continuously and allows you to query data on different aggregation levels, from a raw event (single click) to aggregated calculations spanning multiple months or years. You can use the real-time reporting service either standalone or combined with enricher microservices—for example, to create additional insights, such as average cart value, total order volume, referrer analysis, and so on.

Identity service
All identities of a user are clustered in a single profile by the identity service clusters. It provides service endpoints to create, read, alter, and delete identities, profiles, and their relations.

Foundation Integration
Context-driven services foundation provides several prebuilt integrations, which allow it to set up and use pre-defined scenarios with low effort. It captures and analyzes customer interactions, contexts, and behaviors to create a continually evolving intelligent profile, enabling a deeper understanding of a customer’s motivation and intent in real time. This enables customer-facing touchpoints like commerce and marketing to react and interact in more personal ways. Foundation has prebuilt integrations with SAP Commerce and SmartEdit for personalization and the assisted service module, as follows:

Integration with SmartEdit for personalization
The prebuilt integration of context-driven services foundation with personalization allows for mapping of the customer data from foundation to segments and using it in personalization via SmartEdit. The integration with personalization is based on three extensions:

- The personalizationIntegration extension translates data from foundation to commerce with two mappers, CxScriptProfileMapper to interpret the context-driven services foundation data and CxFixedSegmentsProfileMapper to return fixed segments defined in a segmentation.
- The yaasconfiguration extension provides data models to hold required configurations to communicate with services to retrieve data.
- The personalizationprofile extension has mappers that navigate the foundation data and convert it; it can also resolve product and category affinities.

Integration with SAP Commerce
The prebuilt integration between SAP Commerce and foundation facilitates the data exchange between the two. The system is configured to retrieve the foundation data synchronously during login. Upon user login, an event is triggered that executes the following steps:

- Calls context-driven services foundation to retrieve the user profile
- Maps the context-driven services foundation data to segment data for personalization
- Updates the segments for the users in the database
- Calculates the personalized experience for the storefront

Figure 5.15 depicts the data integration with SAP Commerce to create a meaningful customer experience.

![Figure 5.15 - Sending Data to SAP Commerce System](image-url)
Integration from SAP Commerce

The data from SAP Commerce is sent to foundation for the events like registration/login, page views, adding items to the cart, order placement, and banner clicks. The data is sent through profiletagaddon, the yprofileintegration module, and hybrisanalyticsaddon, as shown in Figure 5.16.

Figure 5.16 Getting Data from SAP Commerce

5.5.2 Segmentation

The aim of segmentation is to provide a personalized shopping experience to the customer when she interacts with the storefront by grouping customers into predefined segments in real time. This functionality is available for anonymous users as well as registered users, aligned with their consent preferences.

Real-time segmentation supports personalization from the customer’s first interactions on site, if she has consented. Business users can quickly understand the segments that are driving various customer experiences and leverage that information to increase conversion. Segmentation functionality helps personalize web storefront content, promotions, and search. Segmentation drives conversion by placing the right products at the right time to deliver customer-specific experiences.

The process begins when the customer begins interactions with the storefront and storefront sends event data to context-driven services foundation. The context-driven services builds the customer profile and, based on the storefront interaction events, assigns the customer to the appropriate segment. The segment data is transferred to SmartEdit for meaningful personalization of the storefront layout. Figure 5.17 summarizes the segmentation process.

Figure 5.17 Segmentation Process in Real World

Let’s look at how to create a segment and how to match customers with the right segments.

Segment Creation

Segment Builder is a tool in which you can define groups to segment customers by different attributes and conditions. The customer data may come from various sources and channels. To begin, follow these steps:

1. In Segment Builder, to add a new segment, click Segmentation and then Segment Builder; this will take you to the screen shown in Figure 5.18.

Figure 5.18 Create New Segment

2. Click the Add button and a window for defining the new segment and conditions will appear, as shown in Figure 5.19.
3. Provide the segment rule details, like Segment Name, Description, and Active period.

4. Define the conditions a customer must meet to be part of this segment.

5. Save the segment and toggle the Active button to activate the segment.

When a customer meets the required criteria, the segment service will place the segment in that customer’s profile.

There are multiple condition types that are available to define a segment. For example, order conditions (like order value, order count, last order value, etc.), cart conditions (cart value, products in the cart, product value in the cart, etc.), customer conditions (customer zip code, gender, brand affinity, locations, etc.), and website conditions (URL matches and funnel level). In addition, custom conditions can be defined as well.

**Figure 5.19** Define New Segment Details and Activate Segment

---

**Segment Calculation**

Context-driven services segmentation ships with a segment calculation service that assigns customers to the segments defined in Segment Builder in real time. It has a number of attributes, as follows:

- As the customer browses the web storefront, the customer profile document is updated by the enricher via the profile service.
- The profile service notifies you of the change in the profile document.
- The segment calculation service gets the segment information from the backend and recalculates the segments.
- The segment calculation service stores the calculated segments in the profile service.

This process continues for recalculation of segments in real time based on the customer’s browsing activity. The segment calculation process is shown in Figure 5.20.

**Figure 5.20** Segment Calculation Process

---

**5.6 Summary**

Customer experience is what expresses your brand’s identity to your customers, leads them through their journey, and leaves them with a lasting impression of what you offer. It’s the bridge between your offerings and your customers, your business, and the people it serves.

SAP Commerce provides multiple toolsets for managing your customer experience, from the overall layout, look, and feel of your site to rules and permissions associated with individual components and slots to the properties and contents of the components themselves. Personalization is built into all these toolsets, but most visibly into
SmartEdit and most robustly into context-driven services, the combination of which can help you craft a fresh, relevant, and effective experience for your customers.

One key outcome of a solid customer experience is, of course, an increase in the volume and value of orders—so in the next chapter, we’ll look at how to manage orders, from acceptance to fulfillment to repeat business.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>17</td>
</tr>
<tr>
<td>Preface</td>
<td>19</td>
</tr>
</tbody>
</table>

## 1 Introduction

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 The Commerce Market</td>
<td>25</td>
</tr>
<tr>
<td>1.1.1 Customer Expectations</td>
<td>26</td>
</tr>
<tr>
<td>1.1.2 Vendor Requirements</td>
<td>27</td>
</tr>
<tr>
<td>1.2 The SAP Commerce Story</td>
<td>30</td>
</tr>
<tr>
<td>1.2.1 The Platform and its Evolution</td>
<td>31</td>
</tr>
<tr>
<td>1.2.2 Modularizing</td>
<td>35</td>
</tr>
<tr>
<td>1.2.3 Cockpits</td>
<td>35</td>
</tr>
<tr>
<td>1.2.4 Cloudification</td>
<td>38</td>
</tr>
<tr>
<td>1.3 Summary</td>
<td>38</td>
</tr>
</tbody>
</table>

## 2 Getting Started with SAP Commerce

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Before You Begin</td>
<td>41</td>
</tr>
<tr>
<td>2.1.1 Application Lifecycle Framework</td>
<td>42</td>
</tr>
<tr>
<td>2.1.2 Technical Requirements and Infrastructure</td>
<td>43</td>
</tr>
<tr>
<td>2.1.3 Methodology and Recommended Practices</td>
<td>45</td>
</tr>
<tr>
<td>2.2 Running Your Project</td>
<td>51</td>
</tr>
<tr>
<td>2.2.1 Defining the Scope</td>
<td>51</td>
</tr>
<tr>
<td>2.2.2 Initiation Phase</td>
<td>54</td>
</tr>
<tr>
<td>2.2.3 Foundation Phase</td>
<td>57</td>
</tr>
<tr>
<td>2.2.4 Development and Engineering</td>
<td>61</td>
</tr>
<tr>
<td>2.2.5 Deployment and Cutover/Go-Live</td>
<td>68</td>
</tr>
<tr>
<td>2.2.6 Application Management</td>
<td>71</td>
</tr>
<tr>
<td>2.3 Building Your Team</td>
<td>72</td>
</tr>
<tr>
<td>2.3.1 Initiation Phase</td>
<td>72</td>
</tr>
</tbody>
</table>
## 2.3.2 Foundation Phase ................................................................. 74
## 2.3.3 Engineering Phase .............................................................. 78
## 2.3.4 Deployment Phase .............................................................. 82
## 2.3.5 Application Management .................................................... 86

### 2.4 Operating in the Cloud ......................................................... 89
#### 2.4.1 Cloud Portal ................................................................. 89
#### 2.4.2 Cloud Readiness Check .................................................... 89
#### 2.4.3 Platinum Upgrade Program .............................................. 90

### 2.5 Upgrade Projects ................................................................. 91
#### 2.5.1 Upgrade Considerations .................................................... 92
#### 2.5.2 Roles and Responsibilities ................................................ 94

### 2.6 Summary ............................................................................. 96

### 3 Commerce Management ...................................................... 97

#### 3.1 Sales Organization ............................................................. 98

#### 3.2 Cart Management ............................................................... 99
##### 3.2.1 Extensions and Configurations .......................................... 100
##### 3.2.2 Cart Merging .............................................................. 101

#### 3.3 Product Recommendation .................................................. 103

#### 3.4 Wishlists and Checkout ....................................................... 106
##### 3.4.1 Wishlists ................................................................. 106
##### 3.4.2 Checkout ................................................................. 109
##### 3.4.3 PCI Compliance ........................................................ 113

#### 3.5 Promotions ........................................................................ 114
##### 3.5.1 Promotional Pricing ...................................................... 115
##### 3.5.2 Promotions Engine ...................................................... 118
##### 3.5.3 Rule Engine .............................................................. 126

#### 3.6 SAP Commerce Accelerator Storefronts .............................. 137
##### 3.6.1 Extensions and Add-Ons ................................................. 138
##### 3.6.2 Storefronts ............................................................... 140
##### 3.6.3 Sample Data and Configurations ...................................... 144

#### 3.7 Summary ............................................................................. 146

### 4 Product Content and Catalog Management ................................ 149

#### 4.1 Product Catalog Management ............................................ 150
##### 4.1.1 Structuring Products and Categories .................................. 151
##### 4.1.2 Catalog Versions ........................................................ 153
##### 4.1.3 Managing Multiple Catalogs .......................................... 154
##### 4.1.4 Synchronizing Catalogs ............................................... 158

#### 4.2 Product Cockpit .................................................................. 159
##### 4.2.1 Getting Started with the Product Cockpit ........................... 161
##### 4.2.2 Creating Categories ....................................................... 174
##### 4.2.3 Creating and Enriching Products ...................................... 178
##### 4.2.4 Synchronizing Products ................................................ 184

#### 4.3 Configurable Products ....................................................... 185

#### 4.4 Bundling ............................................................................. 186

#### 4.5 SAP Digital Asset Management .......................................... 191

#### 4.6 Summary ............................................................................. 193

### 5 Customer Experience Management ........................................ 195

#### 5.1 Page Templates for All Channels ........................................ 196

#### 5.2 Web Content Management System .................................... 202

#### 5.3 SmartEdit .......................................................................... 209
##### 5.3.1 Framework and Architecture ........................................ 209
##### 5.3.2 Content Management .................................................. 218
##### 5.3.3 Components ................................................................ 219
##### 5.3.4 Restrictions Management ............................................. 221
##### 5.3.5 Navigation .................................................................... 223
##### 5.3.6 User Accounts and User Groups .................................... 225

#### 5.4 Personalization ................................................................. 226
##### 5.4.1 Personalization Architecture ......................................... 227
##### 5.4.2 Configuring Personalization .......................................... 228
##### 5.4.3 Extending Personalization ............................................. 229
##### 5.4.4 Advanced Personalization Module .................................. 230
6 Order Management

6.1 Sales Organization
   6.1.1 General Tab
   6.1.2 Employees Tab
   6.1.3 Accounts Tab
   6.1.4 Assisted Service Module Customer List

6.2 Quotation Management
   6.2.1 Quotation Lifecycle
   6.2.2 Requoting Inactive Quotes
   6.2.3 Viewing Quotes
   6.2.4 Email Notifications

6.3 Capturing an Order
   6.3.1 Cart and Checkout
   6.3.2 Store Locator
   6.3.3 Buy Online, Pick Up in Store
   6.3.4 Payment and Taxes

6.4 Distributed Order Management
   6.4.1 Order Fulfillment Cockpit
   6.4.2 Stock and Availability
   6.4.3 Enterprise Inventory
   6.4.4 Endless Aisle

6.5 Order Orchestration
   6.5.1 Warehouse and Points of Service
   6.5.2 Business Processes and Order Status
   6.5.3 Order Splitting and Consignments
   6.5.4 Sourcing and Allocation to Fulfillment Locations
   6.5.5 Orchestrating Multiple Backends

5.5 Context-Driven Services
   5.5.1 Foundation
   5.5.2 Segmentation

5.6 Summary

6.5.5 Orchestrating Multiple Backends

6.6 Fulfillment Execution
   6.6.1 External Fulfillment
   6.6.2 Pick, Pack, and Ship
   6.6.3 Pick Up in Store
   6.6.4 Declining or Relocating Consignments
   6.6.5 Task Assignment Workflow: Inbox

6.7 Order Lifecycle
   6.7.1 Backoffice Customer Support Cockpit
   6.7.2 Additional Order Actions
   6.7.3 Order Cancellations
   6.7.4 Order Returns
   6.7.5 Order Management Self-Service

6.8 Summary

7 Support and Service

7.1 Backoffice Customer Service Cockpit
   7.1.1 Administration and Unbound Customer Attributes
   7.1.2 Product Ratings and Reviews
   7.1.3 Customer Account Information
   7.1.4 Personal Data Reports
   7.1.5 Orders
   7.1.6 Returns
   7.1.7 Support Ticket Management

7.2 Assisted Service Module
   7.2.1 Customer 360° View
   7.2.2 Order Placement Support
   7.2.3 Promotions
   7.2.4 Integrating with Context-Driven Services

7.3 Self-Service

7.4 Summary
11 Integration

11.1 Basics ........................................................................................................................................ 452
11.2 Master Data Management ........................................................................................................ 454
11.2.1 Data Models and Objects Overview ............................................................................... 454
11.2.2 Customer Data Integration ............................................................................................ 456
11.2.3 Product Data Integration .................................................................................................. 463
11.2.4 Pricing Data Integration ...................................................................................................... 464
11.3 Order Management Scenarios ................................................................................................. 467
11.3.1 Asynchronous Order Management ................................................................................. 467
11.3.2 Synchronous Order Management .................................................................................... 470
11.3.3 Leveraging Order Management for Commerce ............................................................ 471
11.3.4 Multibackend Integrations ............................................................................................... 473
11.4 Price Calculation in an Integrated SAP Landscape ................................................................. 475
11.4.1 Price Calculation Based on Consistent Master Data ....................................................... 475
11.4.2 Price Calculation Based on SAP Pricing Engine ............................................................. 475
11.5 Integration with SAP Marketing Cloud ..................................................................................... 476
11.5.1 Data Replication to SAP Marketing Cloud ........................................................................ 478
11.5.2 Product and Offer Recommendations (B2C) .................................................................... 478
11.5.3 Personalization with Segmentation .................................................................................... 480
11.5.4 Clickstream Data Transfer to SAP Marketing Cloud ....................................................... 482
11.6 Integration with SAP Sales Cloud and SAP Service Cloud ..................................................... 482
11.7 Integration with SAP Subscription Billing .................................................................................. 485
11.8 Integration with SAP Customer Engagement Center ............................................................... 486
11.9 Integration with the SAP Digital Payments Add-On ................................................................. 487
11.10 Integration with SAP Jam Communities .................................................................................. 488
11.11 Integration Strategy and Technology ...................................................................................... 490
11.11.1 SAP Commerce, Data Hub ............................................................................................. 490
11.11.2 SAP Cloud Platform Integration and Integration Objects .......................................... 492
11.12 Summary .................................................................................................................................. 493

12 Cloud Portal ................................................................................................................................ 495

12.1 Getting Started ............................................................................................................................... 496
12.1.1 Portal Access ....................................................................................................................... 496
12.1.2 Code Repository Setup ....................................................................................................... 497
12.1.3 Building Manifests ............................................................................................................. 499
12.2 SAP Commerce Cloud Operations ............................................................................................. 499
12.2.1 Deployment ......................................................................................................................... 500
12.2.2 Configuration ..................................................................................................................... 503
12.2.3 User Management .............................................................................................................. 509
12.3 Infrastructure Services ................................................................................................................ 510
12.3.1 Scalability .......................................................................................................................... 511
12.3.2 Monitoring ........................................................................................................................ 511
12.3.3 Maintenance ..................................................................................................................... 512
12.3.4 Database Services ............................................................................................................ 512
12.4 Security ...................................................................................................................................... 514
12.5 Support ..................................................................................................................................... 516
12.5.1 User Creation ..................................................................................................................... 516
12.5.2 How to Create an Incident ................................................................................................. 517
12.6 Summary .................................................................................................................................. 517

The Authors ...................................................................................................................................... 519
Index .................................................................................................................................................. 521
Index

A
Abstract pages ........................................................ 204
Abstraction layer ..................................................... 127
Acceptance test-driven development .......................... 136
Access policies ....................................................... 438
Access rights ......................................................... 391, 439
Accounts ............................................................... 247
Actions .................................................................. 127, 134
Adaptive design ...................................................... 196
Adaptive search ..................................................... 368, 369, 378, 383
Add-ons ................................................................. 139, 421
Addresses ............................................................... 458, 459
Administration Cockpit ......................................... 261, 456, 459
Administration console ........................................... 391, 445
Administration perspective .................................... 37
Administrators ....................................................... 306, 307, 391
Adobe Experience Manager .................................. 139
Aggregated view ................................................... 166
AJAX calls ............................................................. 217
AngularJS module ................................................. 213
Anonymous user .................................................... 390
API hooks ............................................................. 100
Apparel store ........................................................ 140, 152, 196, 408
Application lifecycle framework ............................... 42
Application Link Enabling (ALE) .............................. 454
Application management ........................................ 71
Architecture .......................................................... 59, 419
layers .................................................................. 420
Assets .................................................................. 191
Assisted service agent .......................................... 334, 335
Assisted service module (ASM) ............................... 249, 304, 334, 335, 341
customer list ....................................................... 248
header .................................................................. 338
session ................................................................. 339
Assortment view ................................................... 167, 174
ATP Formula Service API ....................................... 264
Attribute permissions .......................................... 438
Authorizations ....................................................... 448, 517
Autocomplete ....................................................... 357

B
Backoffice .............................................................. 36, 37, 104, 115, 116, 124, 134, 174, 283, 354, 384, 427
workflows ............................................................. 431
Backoffice Product Cockpit ..................................... 150, 160, 161, 179, 371, 385, 404
Backup process ..................................................... 70
Beans .................................................................. 230, 343, 414
Behavioral targeting groups .................................... 230, 232
Bidirectional contact maintenance ......................... 458
Billbucket ............................................................... 497, 498
Blacklist ................................................................. 442
Boolean scoring ..................................................... 359
Boost and bury ..................................................... 381
multiplier ............................................................. 378
Boost rules ........................................................... 369
Brick-and-mortar storefront ................................. 26
Browser area ........................................................ 164
Build manifests ..................................................... 499, 507
Bundling ............................................................... 186, 187
criteria ................................................................. 188
pop-up ................................................................. 420
template ............................................................. 190
template components ......................................... 188
Business execution ............................................... 42
Business logic layer .............................................. 420
Business-to-business (B2B) .................................. 26, 57
business units ..................................................... 396, 397
customer data ..................................................... 457
customers ........................................................... 396
units .................................................................. 458, 460
Business-to-business-to-customer (B2B2C) ............... 141
Business-to-customer (B2C) .................................. 97, 461

Automated scaling .............................................. 511
Automation ......................................................... 30
Available-to-promise (ATP) ................................. 262, 263, 469
check ................................................................. 472
formula .............................................................. 264
Azure database service ......................................... 512
Model-view-controller 138, 209, 400
module gen tool 35
Modules 35
Monitoring 511, 513
Multicore coupons 125
Multicountry rollouts 407

N
Navigation 208, 223, 353
Navigation nodes 354
Nonfunctional requirements 59

Q
Objects 454
OData 492, 493
Omni Commerce Connect 139, 256
Omni channel commerce 195
One-time consumer 461
Online commerce 26
Open Web Application Security Project (OWASP) 447
Order calculations 135
Order confirmation 109
Order Fulfillment Cockpit 260, 261, 263, 265, 268, 275, 283, 294
credentials 261
inbox 282
Order history 286, 297
Order management 243, 269, 274, 467
asynchronous 467, 472
for SAP Commerce 471
My Account 297
real-time asynchronous 468
self-service 296
synchronous 470
Orders 256, 283, 318, 348, 456
actions 288
business process 289
business process
cancellations 258, 290, 299
details 284
detail
history 318, 320
lifecycle 283
orchestration 266
place on hold 288

Orders (Cont.)
placement support 340
recent rules 231
splitting 257, 269
status 269
versioning 257

P
Page templates 196, 205
Pages 204
Partial return 325
Password security 442
Payment gateways 114
Payment information 109
Payments 260, 285, 320
PayPal Express 114
PCI compliance 113
Permissions 307, 310, 334, 394, 436
assignment 439
best practices 440
services 439
Personal data 443
Personalization 226, 307, 480
advanced architecture 230
configuration 228
extension 227
extensions 229
Persons 59
definition 60
Perspectives 36
Pick, pack, and ship 277
Pixel counts 196
Plain old Java objects (POJOs) 128
Platinum upgrade program 90
Points of service 267
Porter algorithm 365
Power tools store 140, 152
Presentation layer 420
Prices 100, 402, 464
calculation 468, 475
conditions 189, 465
conversion 404
engine 475

Prices (Cont.)
lists 396
Process integration 452
Product attributes 356, 358
weighting 359
Product browser toolbar 166
Product Catalog Cockpit 184
Product catalogs 141, 208
Product Cockpit 104, 416
perspective 384
Product content management 31
Product data model 33
Product detail page 350, 416
Product master 464
Product master replication 463
Product pages 204, 206
Product perspective 168
grid view 171
list view 172
qualifying tools 170
Product ratings 309
Product recommendations 103, 225, 476, 478, 479
replaces 105
Product references 104
Product reviews 478
Product variants 153
Product view 162
Product visibility 153
Products 149, 343
attributes 153
bulk edit 181
content 167
create manually 182
disable rules 182
edit 181, 182
enrich 178
grouping 151
hierarchies 152
packages 186
price 189
search 355
set 188
structure 151
synchronize 184
variants 185

Profile service 236
Profile tag 235
Project delivery frameworks 42
Project documentation 45, 61
Project management 47
Project planning 61
Promote product 381
Promotion engine 114
abstract rules 119
Promotions 100, 114, 319, 337, 342
business cases 115
clone rule 123
conditions 121
drools 128
functionality 120
messages 122
planning 116, 124
pricing 115
qualifying tools 122
rules 121
technical overview 118
templates 120
Property value provider 373
Public APIs 490

Q
Qualification 55
Quality management 259
Queries 358
conditions 380
syntax 362
Quotations 483
Quotes 249
details 252
lifecycle 249
statuses 252
threshold 249
view 252
Quote-to-cash 256

R
RACI 52
Read permissions 437
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH URL</td>
<td>497</td>
</tr>
<tr>
<td>SSL certificates</td>
<td>441, 514, 515</td>
</tr>
<tr>
<td>SSL handshake</td>
<td>441</td>
</tr>
<tr>
<td>Stemming</td>
<td>365</td>
</tr>
<tr>
<td>Stock information</td>
<td>258</td>
</tr>
<tr>
<td>Stock levels</td>
<td>262, 268</td>
</tr>
<tr>
<td>Stop words</td>
<td>364, 376</td>
</tr>
<tr>
<td>Store locator</td>
<td>259</td>
</tr>
<tr>
<td>Subscription order replication</td>
<td>485</td>
</tr>
<tr>
<td>Subscription product replication</td>
<td>485</td>
</tr>
<tr>
<td>Subtypes</td>
<td>33</td>
</tr>
<tr>
<td>Summary view</td>
<td>165</td>
</tr>
<tr>
<td>Support tickets</td>
<td>326, 332, 349, 487</td>
</tr>
<tr>
<td>attributes</td>
<td>330</td>
</tr>
<tr>
<td>display</td>
<td>333</td>
</tr>
<tr>
<td>management</td>
<td>328</td>
</tr>
<tr>
<td>priority</td>
<td>331</td>
</tr>
<tr>
<td>properties</td>
<td>328</td>
</tr>
<tr>
<td>unassigned</td>
<td>331</td>
</tr>
<tr>
<td>S-user</td>
<td>510</td>
</tr>
<tr>
<td>Synonyms</td>
<td>375</td>
</tr>
<tr>
<td>directions</td>
<td>376</td>
</tr>
<tr>
<td>System administrator role</td>
<td>509</td>
</tr>
<tr>
<td>Type permissions</td>
<td>437</td>
</tr>
<tr>
<td>Type system</td>
<td>33</td>
</tr>
<tr>
<td>Upgrade projects</td>
<td>91</td>
</tr>
<tr>
<td>roles and responsibilities</td>
<td>92</td>
</tr>
<tr>
<td>User Datagram Protocol (UDP)</td>
<td>242</td>
</tr>
<tr>
<td>User groups</td>
<td>146, 225, 343, 392, 394, 438</td>
</tr>
<tr>
<td>User management</td>
<td>509</td>
</tr>
<tr>
<td>User rights</td>
<td>436</td>
</tr>
<tr>
<td>Users</td>
<td>390</td>
</tr>
<tr>
<td>accounts</td>
<td>225</td>
</tr>
<tr>
<td>enabling</td>
<td>436</td>
</tr>
<tr>
<td>Workflows</td>
<td>432</td>
</tr>
<tr>
<td>backoffice</td>
<td>431</td>
</tr>
<tr>
<td>creation</td>
<td>433</td>
</tr>
<tr>
<td>execution</td>
<td>434</td>
</tr>
<tr>
<td>items</td>
<td>434</td>
</tr>
<tr>
<td>manage templates</td>
<td>431</td>
</tr>
<tr>
<td>templates</td>
<td>430, 432</td>
</tr>
<tr>
<td>types</td>
<td>430</td>
</tr>
<tr>
<td>Workshops</td>
<td>58</td>
</tr>
<tr>
<td>yForms</td>
<td>143, 144</td>
</tr>
</tbody>
</table>
Seema Vishnoi is the head of SAP Commerce market development and go-to-market activities in Latin America. She has played key roles in solution management, sales operations, and innovative business solutions and has successfully lead various CRM initiatives for customers within SAP. Outside of SAP, Seema is an adjunct professor at Villanova University.

Sietse Verhoog is the director of SAP Customer Experience Expert Services for Latin America and the Caribbean at SAP. He is responsible for the SAP C/4HANA experts, architects, developers, and industry specialists based out of Mexico, Colombia, and Brazil who provide support to the SAP C/4HANA customers and partners throughout the region.

Drew Messinger-Michaels co-created the CX Business Consulting and Functional Consulting groups within SAP Commerce, along with SAP Customer Experience Expert Services offerings such as the Commerce Capabilities Workshop, Solr Workshop, Roles and Permissions Workshop, Hybris Health Check, and Functional Analyst Certification.

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.