Fast Path to B2C Commerce Developer Certification

Module 5: Page Designer and Job Framework

Module 5.1: Page Designer

Objective - By the end of this module you will be able to **define** a component type, and **use it** on a page

This module will cover:

- Pages, regions and components
- Requirements for a new component type
- Component type definition (JSON)
- Component type script (JS)
- Component type template (ISML)

• Why does this functionality matter for implementations?

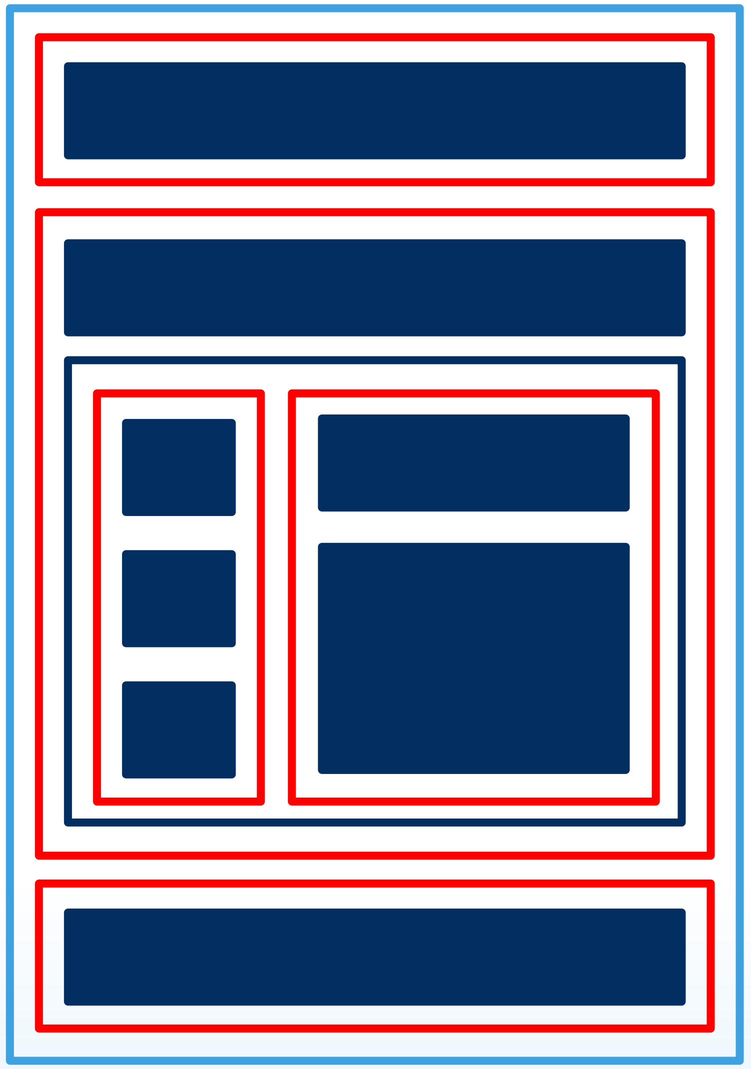
- Page Designer is a flexible content authoring tool for merchants
- Page and component types can be reused on many sites

Page Designer General Terms

Pages, regions and components

Pages are set up in a hierarchical structure:

- **Pages**
  - The outermost container of the structured content
  - Contain regions
  - Constrained by customer group and/or date range
- **Regions**
  - Allow for a hierarchical structuring of components
  - Contain components
- **Components**
  - Have attributes that specify the actual content
  - Constrained by customer group and/or date range
  - Can also be layouts: contain regions that contain other components



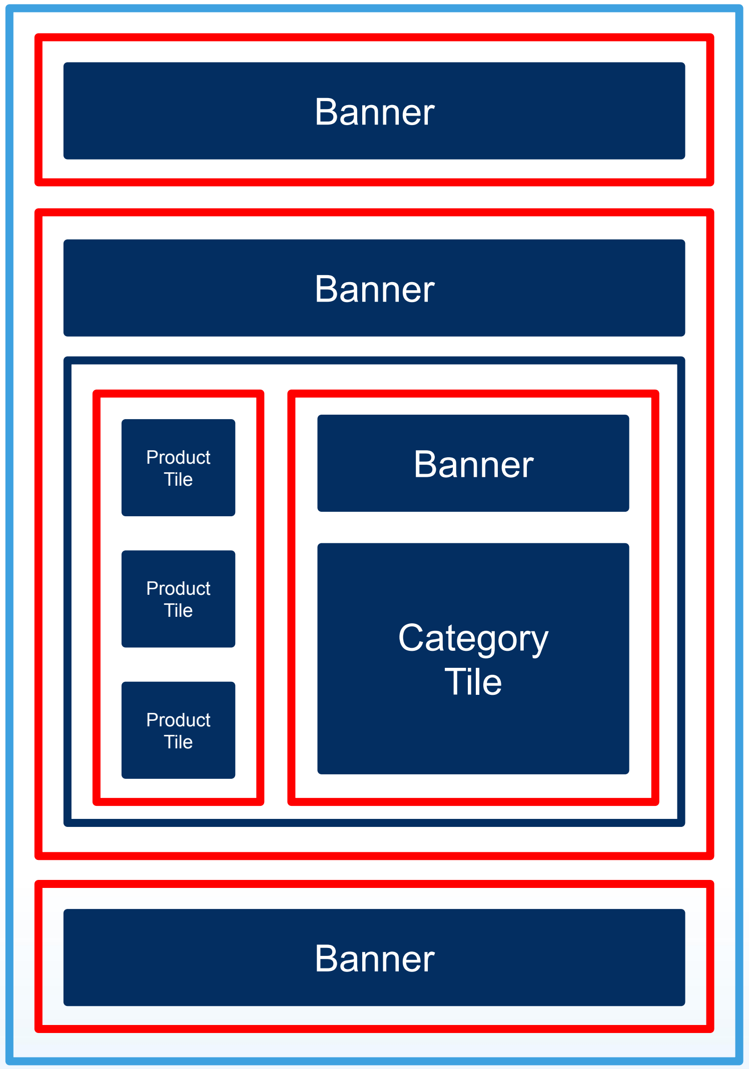
Page Designer Types

Front End Developer

Developers create page types and component types based on project requirements:

- **Page Types**
  - Can also be considered “templates” (promo page, landing page, content page).
  - Contains set regions with specified components available for use in that region.
- **Component Types**
  - The type of content available for use in a specific region (banners, product tiles, category tiles, component layouts, etc).

Promo Page Type



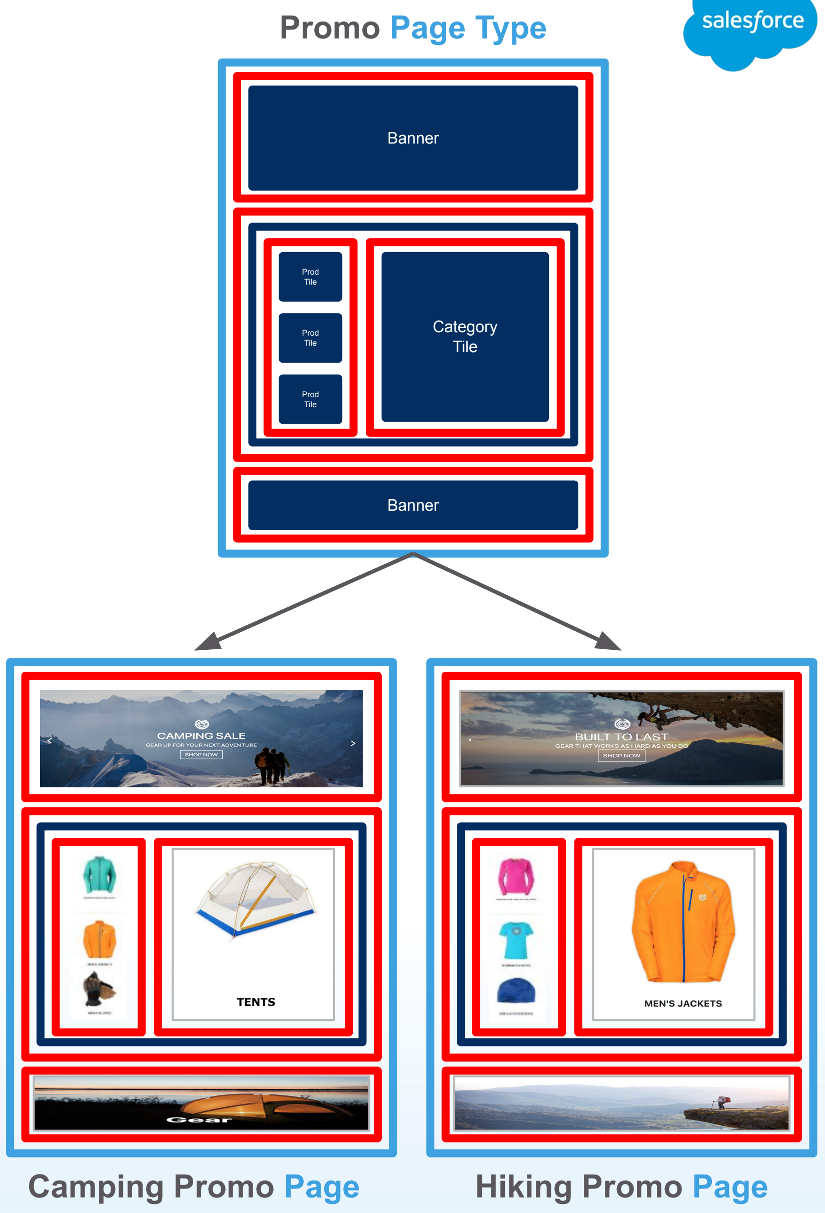
Pages and Components

Web Content Manager

Content managers create the **page** in Business Manager and place **components** in **regions** based on the **page type** and **component types** the developer has made available:

- They also configure a set of attributes required to render the **component**, for example, entering text and selecting an image to be displayed on a banner **component**.
- They can use the same **page types** and **component types** to create multiple new **pages** and won't have to go back to development until they need changes or new **page types** and **component types**.

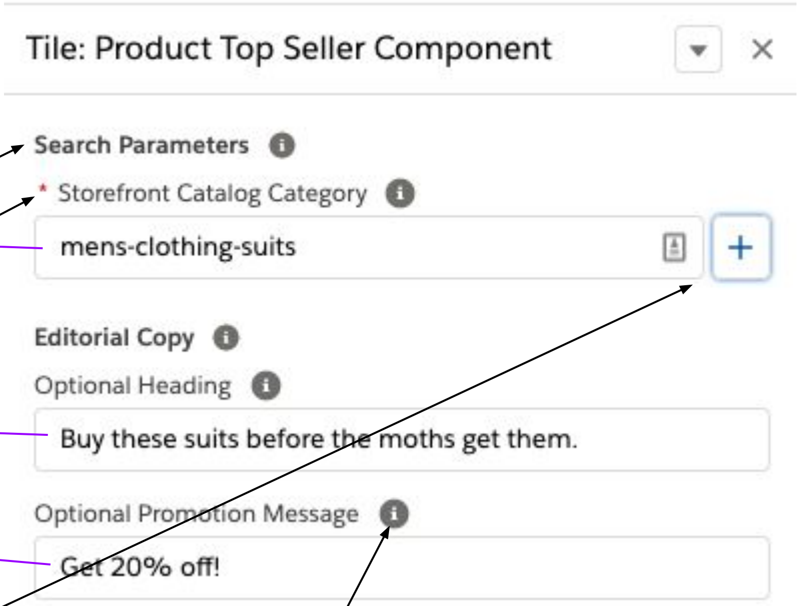
Helpful Hint: Pages and components are instances of page types and component types.



Requirements for a component type: Product Top Seller

Let's capture some requirements

Derived top seller for chosen category


Tile: Product Top Seller Component

Search Parameters

* Storefront Catalog Category: mens-clothing-suits

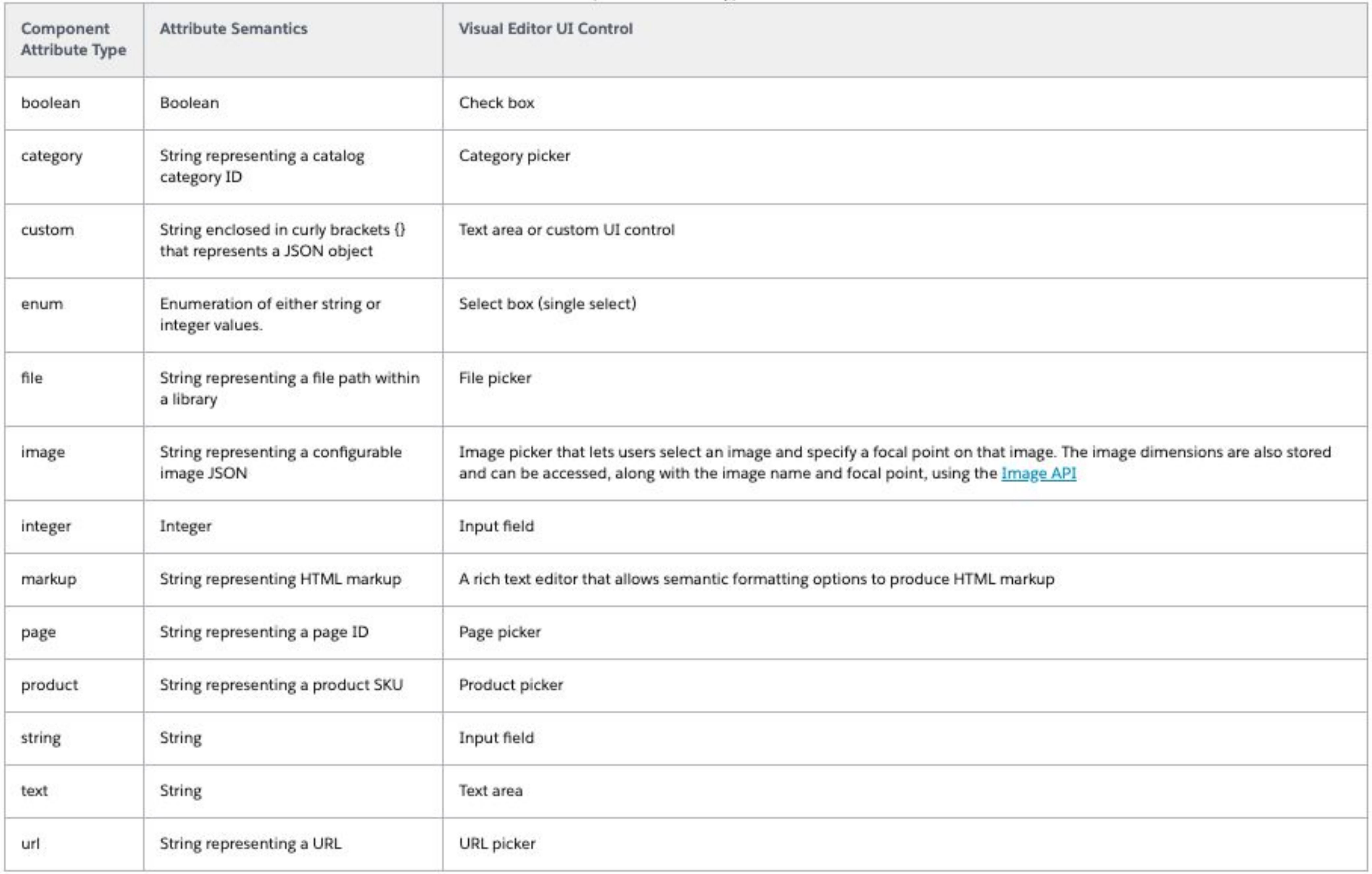
Editorial Copy

Optional Heading: Buy these suits before the moths get them.

Optional Promotion Message: Get 20% off!

Attribute Group / Attribute Name	Internal Attribute ID	Attribute Type	Required?	Attribute Description / Help Guidance
Search Parameters		**Attribute Definition Group**		**Please select a category to search for a top seller.**
Storefront Catalog Category	category	category	TRUE	Please select a storefront catalog category to get its top selling product.
Editorial Copy		**Attribute Definition Group**		**Please specify the editorial copy that will be displayed to customers when this tile is published.**
Optional Heading	headline	Text	FALSE	Optionally, specify text copy to include in the tile display. The default is the product name.
Optional Promotion Message	promotion_message	Text	FALSE	Optionally, specify the promotion text (i.e. message and coupon code) to show below the heading.

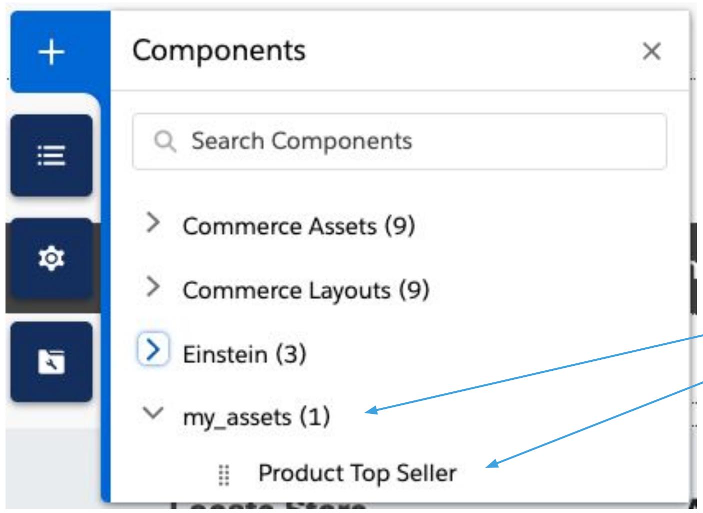
Available attribute types (click to navigate to documentation)



Component Attribute Type	Attribute Semantics	Visual Editor UI Control
boolean	Boolean	Check box
category	String representing a catalog category ID	Category picker
custom	String enclosed in curly brackets {} that represents a JSON object	Text area or custom UI control
enum	Enumeration of either string or integer values.	Select box (single select)
file	String representing a file path within a library	File picker
image	String representing a configurable image JSON	Image picker that lets users select an image and specify a focal point on that image. The image dimensions are also stored and can be accessed, along with the image name and focal point, using the Image API
integer	Integer	Input field
markup	String representing HTML markup	A rich text editor that allows semantic formatting options to produce HTML markup
page	String representing a page ID	Page picker
product	String representing a product SKU	Product picker
string	String	Input field
text	String	Text area
url	String representing a URL	URL picker

Co

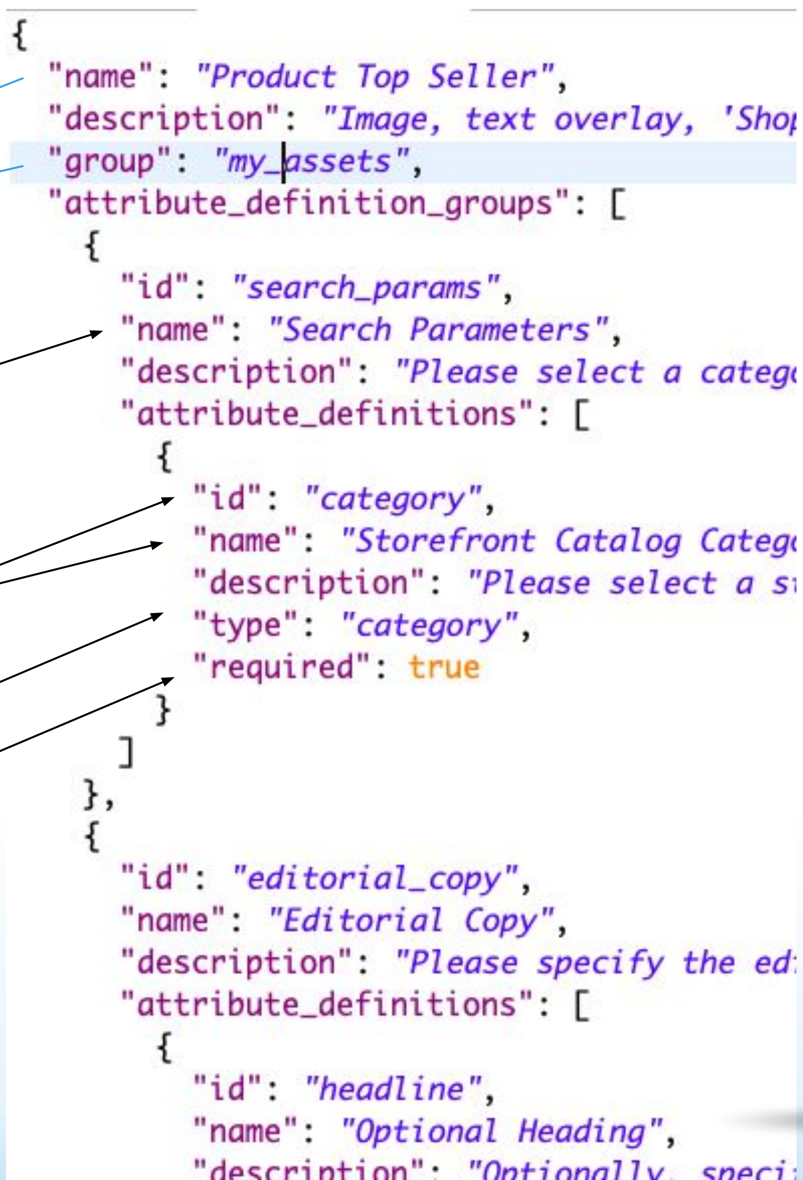
Creating a component-type definition (JSON file)


Components

Search Components

- Commerce Assets (9)
- Commerce Layouts (9)
- Einstein (3)
- my_assets (1)
  - Product Top Seller

Attribute Group / Attribute Name	Internal Attribute ID	Attribute Type	Required?	Attribute Description / Help Guidance
Search Parameters		**Attribute Definition Group**		**Please select a category to search for a top seller.**
Storefront Catalog Category	category	category	TRUE	Please select a storefront catalog category to get its top selling product.


```
{
  "name": "Product Top Seller",
  "description": "Image, text overlay, 'Sho
  "group": "my_assets",
  "attribute_definition_groups": [
    {
      "id": "search_params",
      "name": "Search Parameters",
      "description": "Please select a catego
      "attribute_definitions": [
        {
          "id": "category",
          "name": "Storefront Catalog Catego
          "description": "Please select a s
          "type": "category",
          "required": true
        }
      ]
    },
    {
      "id": "editorial_copy",
      "name": "Editorial Copy",
      "description": "Please specify the ed
      "attribute_definitions": [
        {
          "id": "headline",
          "name": "Optional Heading",
          "description": "Optionally, speci
```

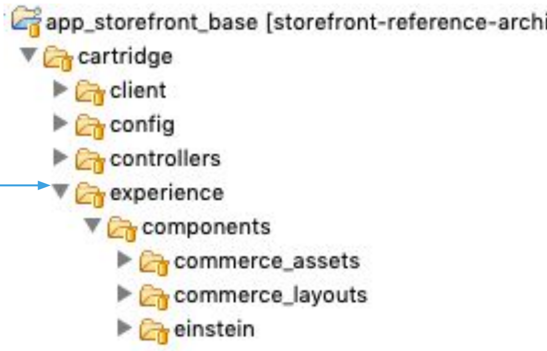
Creating a component-type rendering script (JS file)

A few rules. Consult the documentation for complete description.

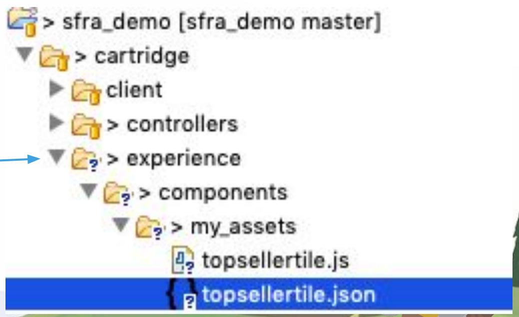
- The script file must:
  - Have the same name as the type's meta definition file, but with a .js extension.
  - Be located in the same directory as the meta definition file
  - Include a render function that returns the markup for the page
  - Pass a HashMap containing the data (model) for the ISML template
  - Use the dw.util.Template API to render the ISML template
- The render function has a context parameter: context.content represents the attributes defined in the meta definition file, but containing the attribute values entered in BM.
- Use this syntax to pass the model to the ISML template:

```
new Template('experience/components/my_assets/topsellertile').render(model).text;
```

- In SFRA, rendering components are located in
- In your cartridge, you can define a similar structure with your own directory to make it easier to identify your own components



```
app_storefront_base [storefront-reference-archi
▼ cartridge
  ▶ client
  ▶ config
  ▶ controllers
  ▼ experience
    ▼ components
      ▶ commerce_assets
      ▶ commerce_layouts
      ▶ einstein
```

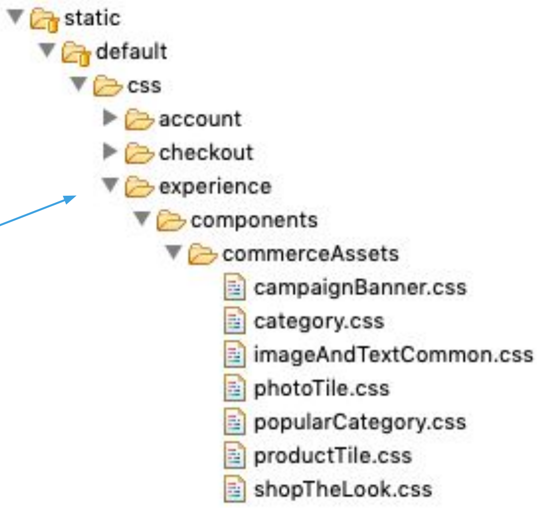


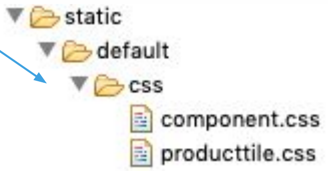
```
> sfra_demo [sfra_demo master]
▼ > cartridge
  ▶ client
  ▶ > controllers
  ▼ > experience
    ▼ > components
      ▼ > my_assets
          topsellertile.js
          topsellertile.json
```

Creating a component-type rendering template (ISML)

Very similar to any other ISML you used before: pdict lives on!

- The rendering template must:
  - Import assets and css to format html
  - Refer to attributes in the model as pdict.attribute
  - Use boolean attributes to determine areas to show
  - Use existing css file or define its own
- CSS files are located in the static folder:
  - In SFRA base, the CSS is compiled from SCSS
  - You can specify your own CSS in your custom cartridge that will be used on your components
- In this webinar, we will use our own css to show how it is different from what comes with SFRA


```
▼ static
  ▼ default
    ▼ css
      ▶ account
      ▶ checkout
      ▼ experience
        ▼ components
          ▼ commerceAssets
              campaignBanner.css
              category.css
              imageAndTextCommon.css
              photoTile.css
              popularCategory.css
              productTile.css
              shopTheLook.css
```


```
▼ static
  ▼ default
    ▼ css
        component.css
        producttile.css
```

Module 5.1 Demo: Creating the Product Top Seller component

1. We will walk thru a component defined in sfra_demo cartridge
2. We will look at:
   a. topsellertile.json
   b. topsellertile.js
   c. topsellertile.isml
   d. topsellertile.properties
3. Use this new component in an SFRA page
   a. Create in default locale
   b. Create for a different locale, and enable that locale
4. Questions:
   a. What method must the rendering script have?
   b. In the ISML template, how do you access the model passed from the script file?
   c. Can the component content be localized by the merchant?
   d. Can the component be used on another site?

Module 5.2: Jobs Framework

- Objective - By the end of this module you will be able to a **create** a job that uses a pre-defined step and a custom step, **run** the job from BM**,** and **locate** the file generated
- This module will cover:
  - Job creation: steps, scheduling and notification
  - Use of steptypes.json and related script to define a custom step
  - Location of IMPEX folder where files are generated
- Why does this functionality matter for implementations?
  - Jobs are a requirement in every implementation
  - A job can use a custom step that is defined via script (no pipelines needed)
  - Jobs can be scheduled to run, and have notification settings

Jobs Framework

The 4th iteration on creating jobs on the platform!

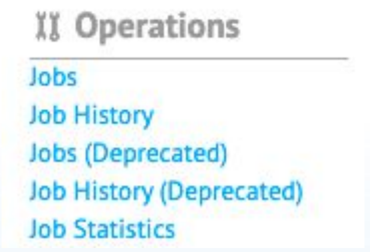
The Jobs Framework allows the platform to perform processes for integration purposes:

- Export orders as a file to an OMS
- Import inventory lists from an OMS
- Import products, and prices from a PIMS
- Call a REST API to retry orders that did not export (yep, a job can do that)
- Export custom objects, and clean up after export

The framework has evolved since the Demandware days, but there are many customers who still use the legacy frameworks!

Instances support both the new Jobs Framework as well as a deprecated version

There is extensive documentation on Jobs.

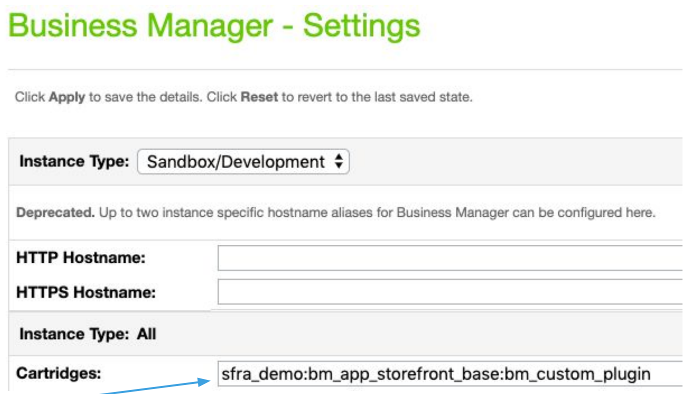


Creating a Job

The mainframers are back!

Main considerations when creating a job:

- Define steps: what is sequential vs. parallel
- Use pre-defined steps that the framework offers as much as possible
- Create custom steps only when needed
- If using a custom step, don't forget to add the cartridge that contains the step to the Business Manager cartridge path!
- When debugging, choose the Business Manager site (Sites-Site), not a storefront site!



Business Manager - Settings

Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

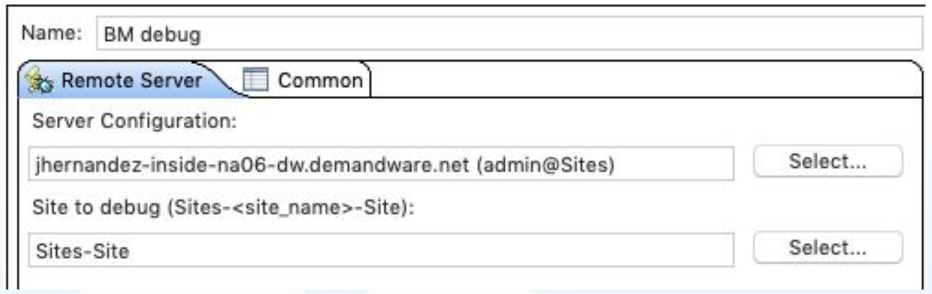
Instance Type: Sandbox/Development

Deprecated. Up to two instance specific hostname aliases for Business Manager can be configured here.

HTTP Hostname:	
HTTPS Hostname:	

Instance Type: All

Cartridges: sfra_demo:bm_app_storefront_base:bm_custom_plugin



Name: BM debug

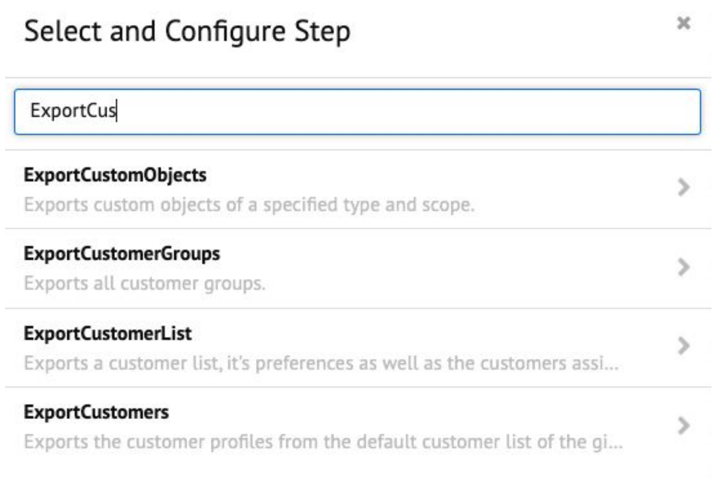
Server Configuration: jhernandez-inside-na06-dw.demandware.net (admin@Sites)

Site to debug (Sites-<site_name>-Site): Sites-Site

Creating a Job Step

Always consider pre-defined steps

Choose from the list of pre-defined steps:


Select and Configure Step

ExportCus

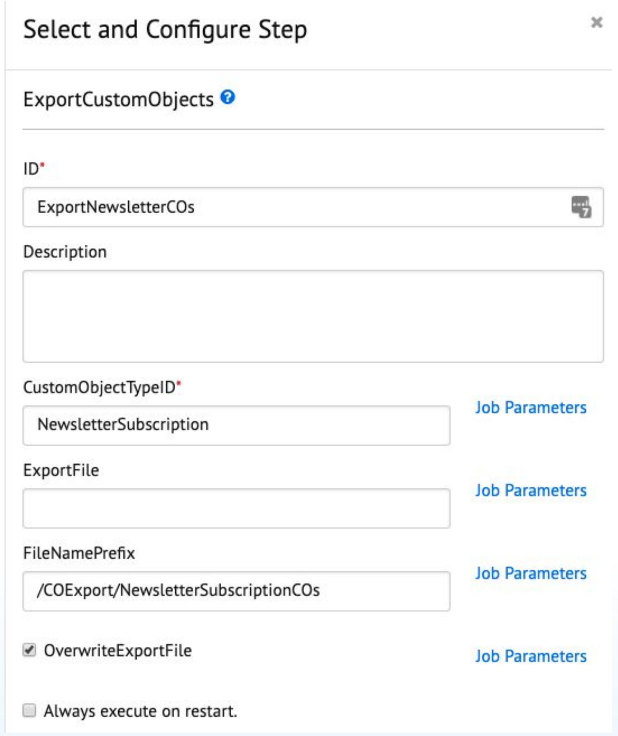
ExportCustomObjects
Exports custom objects of a specified type and scope.

ExportCustomerGroups
Exports all customer groups.

ExportCustomerList
Exports a customer list, it's preferences as well as the customers assi...

ExportCustomers
Exports the customer profiles from the default customer list of the gi...

Configure the step:


Select and Configure Step

ExportCustomObjects

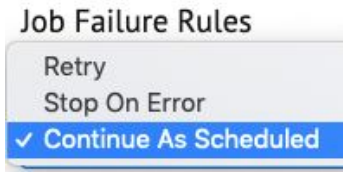
ID*	ExportNewsletterCOs
Description	
CustomObjectTypeID*	NewsletterSubscription
ExportFile	
FileNamePrefix	/COExport/NewsletterSubscriptionCOs

[✓] OverwriteExportFile

[ ] Always execute on restart.

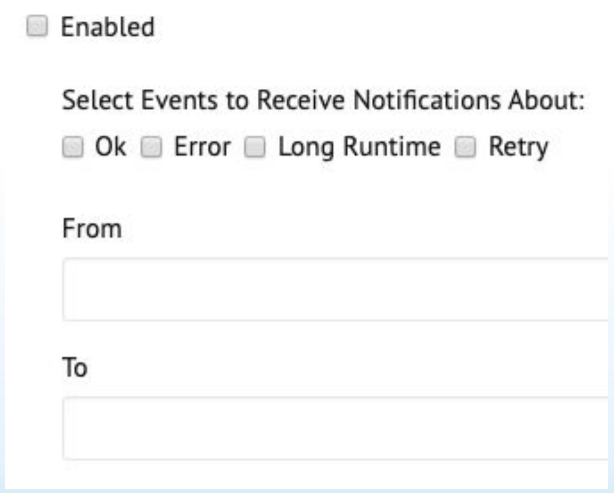
Finalize the job:

- Consider error handling: stop, retry or continue?


Job Failure Rules

Retry
Stop On Error
✓ Continue As Scheduled

- Does anyone care?


[ ] Enabled

Select Events to Receive Notifications About:

[ ] Ok [ ] Error [ ] Long Runtime [ ] Retry

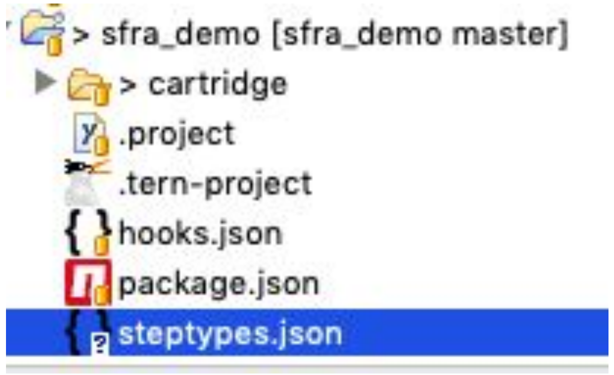
From

To

Creating a Custom Job Step

No need for pipelines anymore!

- Define a steptypes.json file on your cartridge:

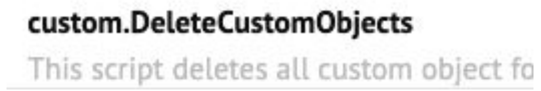


```
> sfra_demo [sfra_demo master]
  > cartridge
  .project
  .tern-project
  hooks.json
  package.json
  steptypes.json
```

- Create a script that implements the function defined in the file above

Add step to the job:

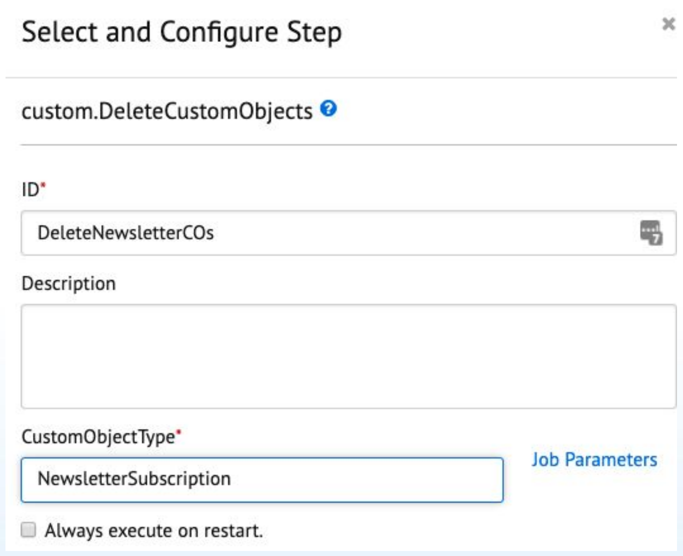
- Find your custom step



custom.DeleteCustomObjects

This script deletes all custom object fo

- Configure the step:



Select and Configure Step

custom.DeleteCustomObjects

ID*: DeleteNewsletterCOs

Description:

CustomObjectType*: NewsletterSubscription

Job Parameters

☐ Always execute on restart.

Run the job:

- Schedules only work in PIG instances, so you must run manually on SBs
- Do you know where your export files go?
  - Development Setup
  - IMPEX dir
- Do you know where you job logs appear?
  - Development Setup
  - Logs dir

Module 5.2 Demo: Create a job that exports and deletes COs

1. We will create a new Job
2. Use a pre-defined step:
   a. Choose a step that exports custom objects to a file
   b. Run the job
   c. Look at the job status: Ok or Error
   d. Look for the file in the IMPEX directory
3. Use a custom step:
   a. Look at the steptypes.json file: script, scope and parameters expected
   b. Look at the script: location, transaction handling
   c. Create a debugging configuration
   d. Add the custom step to the job, after the previous step
   e. Run the job
   f. Step thru the code
   g. Verify all COs are deleted

Module 5.2 Questions

- Can you invoke a job from a storefront URL?
- Do you always have to add a cartridge to the BM cartridge path?
- Can you log errors in a custom step script?
- Why do you have to specify scope when creating a job step?

THANK YOU