Emerald SDK

Service Virtualization Use Cases

Table of Contents

[Rationale 2](#_Toc187423643)

[Service Virtualization 2](#_Toc187423644)

[Software Versions 2](#_Toc187423645)

[Access to Emerald SDK App 2](#_Toc187423646)

[Use Cases 3](#_Toc187423647)

[*‘Flared Accent Chair – Remove Description’* 3](#_Toc187423648)

[*‘Flared Accent Chair - Disable Add To Cart Button’* 3](#_Toc187423649)

[*‘Flared Accent Chair – Out of Stock’* 4](#_Toc187423650)

[*‘Flared Accent Chair –Remove Purple’* 4](#_Toc187423651)

[*‘Flared Accent Chair –Change Price’* 4](#_Toc187423652)

# Rationale

Emerald storefront SDK offers a ‘cut down’, standalone front end modelling an ecommerce store and is extremely useful as a demonstration app for showcasing DevOps processes and tools.

In a normal deployment, Emerald requires access to an HCL Commerce back end which requires a lot of compute to run. The storefront SDK operates in a much reduced footprint and can be run on a laptop in docker without a full ‘back end’. It comprises a React ‘front end’ and mimics back end behaviours using the npm mock framework.

Sadly, the mock framework implementation is incomplete and leaves the user experience somewhat lacking. For example, there are only one or two products in the catalog with product details returned, only one product is in stock that can be added to the shopping cart, and the shopping cart cannot be checked out since there is no implementation to store the shipping addresses etc. Essentially lots of the implementation is missing

# Service Virtualization

One goal of using Service Virtualization with Emerald is to extend the functionality present in Emerald by implementing a more complete ‘back end’ set of functionality.

To begin with, the use cases implemented will be simple, easy to illustrate examples of Service Virtualization in use. Longer term, the hope is that the use of SV will enable a more realistic, complete user experience.

Primarily however, the ability to showcase Service Virtualization with Emerald means we are now able to include Virtual Services in context of DevOps demo’s where Emerald is used. This was not possible until now and instead relied on separate SV demo’s in context of the ‘Vacation booking’ demo app.

For SV demonstrations that are more than surface deep, it is expected that we will continue to use Vacation booking.

# Software Versions

At the time of writing, these examples were built using:

* Emerald 9.1.13-3
* IBM/HCL DevOps Virtualization 11.0.3

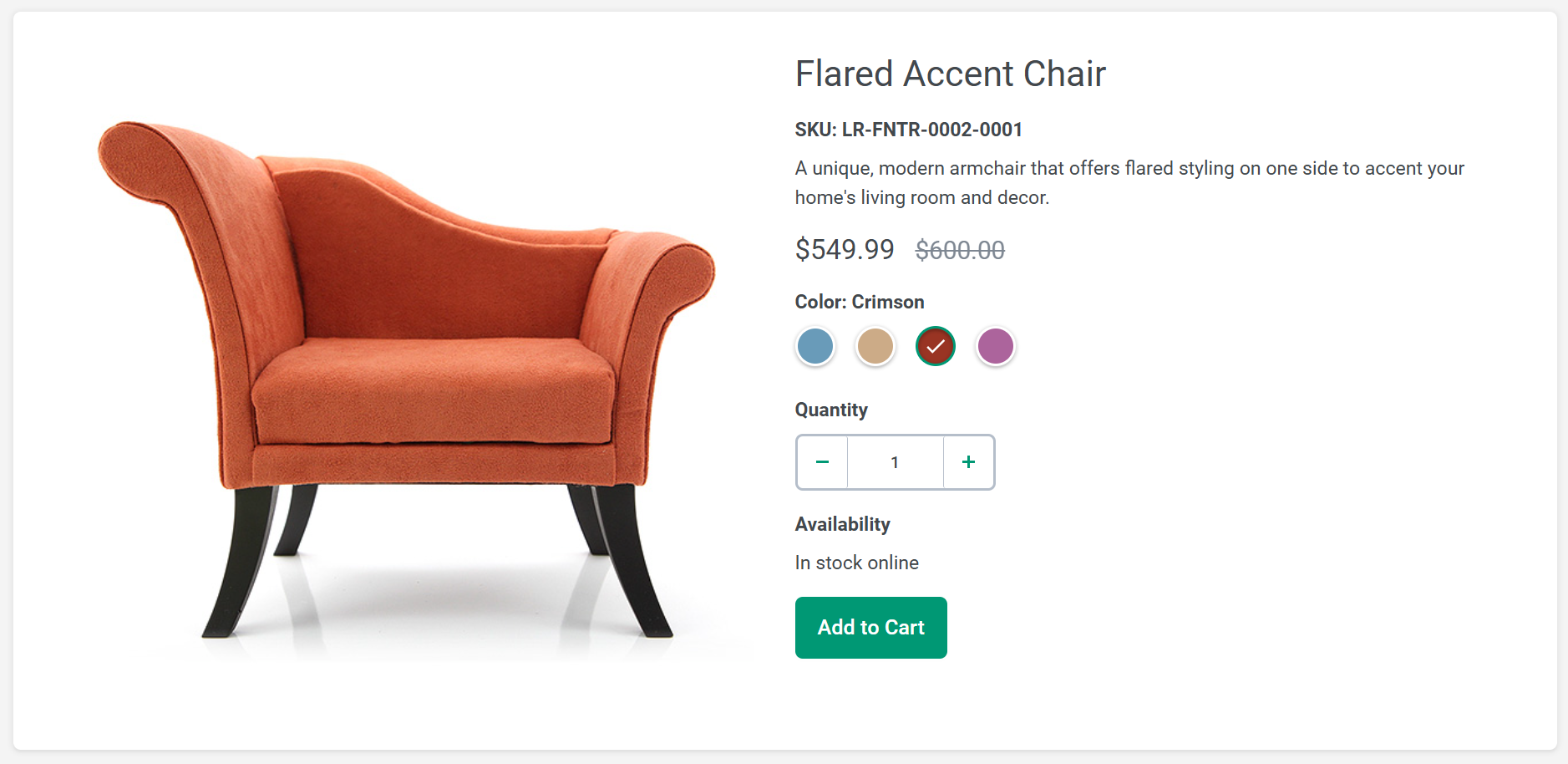
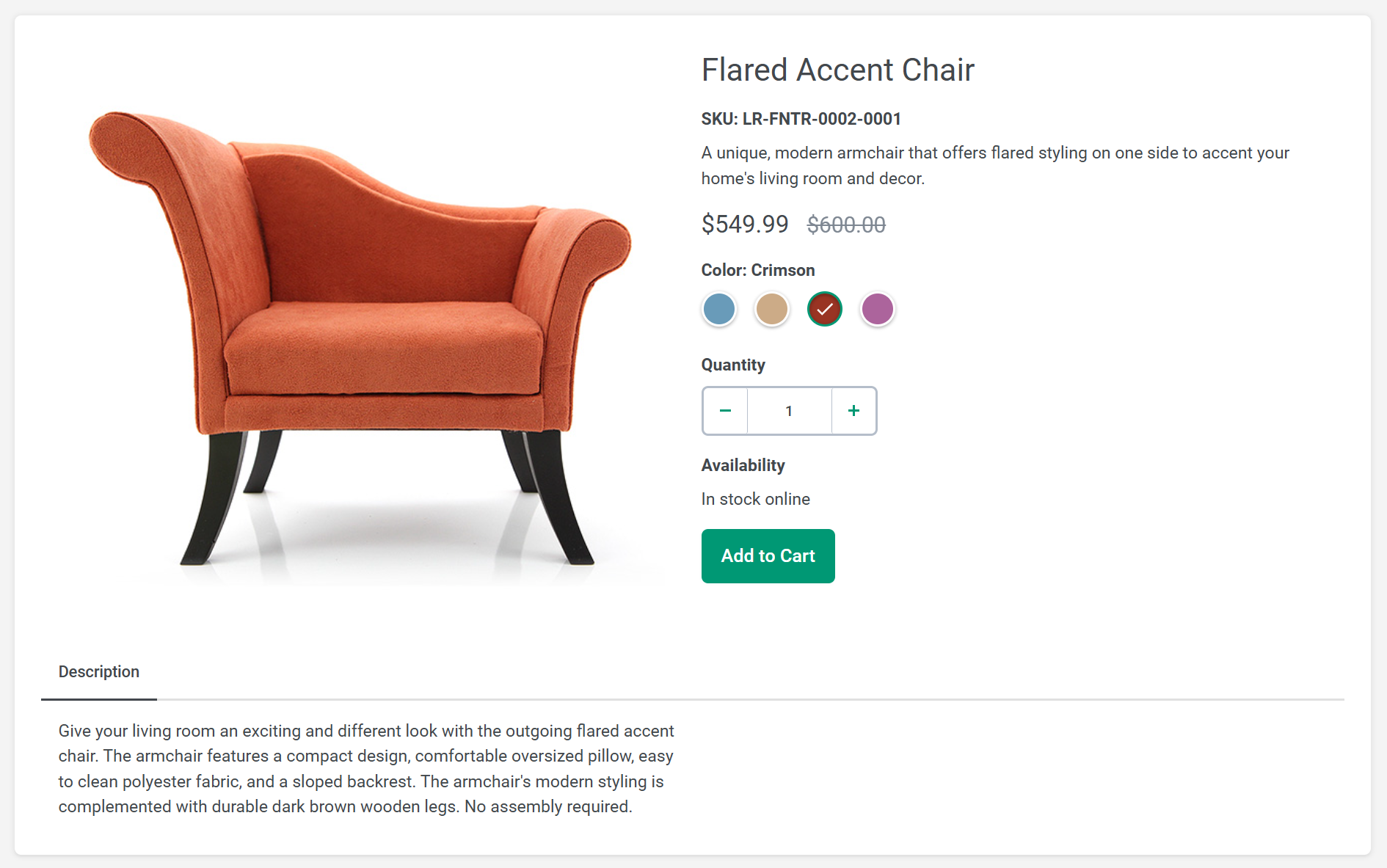
# Access to Emerald SDK App

Emerald is available as a software download from the HCL Commerce channels, however in order to reduce maintenance of these SV assets, we have frozen them at Emerald version 9.1.13-3. A docker container image can be found here <https://github.com/virtualmonster/Emerald_Storefront_App> with instructions on how to build, run and configure the Emerald app along with the pre-installed HTTP Proxy

# Use Cases

### ‘Flared Accent Chair – Remove Description’

Removes the product description text area and label from the product page.



### ‘Flared Accent Chair - Disable Add To Cart Button’

Changes the product merchandisable flag such that the ‘Add to Cart’ button is disabled, despite there being available stock online

A close-up of a chair

Description automatically generated

### ‘Flared Accent Chair – Out of Stock’

Changes the product stock level such that it displays as out of stock and the Add to Cart button is disabled

A close-up of a chair

Description automatically generated

### ‘Flared Accent Chair –Remove Purple’

Removes the option to buy the flared accent chair in purple

A close-up of a chair

Description automatically generated

### ‘Flared Accent Chair –Change Price’

Changes the price of the flared accent chair to $99.99

A close-up of a chair

Description automatically generated

## TO DO

Implement more useful Virtual Services, including:

* Adding inventory for categories other than Furniture, eg add products in the Lighting and Decoration categories
* Investigate whether it is possible to implement the cart functionality to enable a order/checkout process
* Implement the store location service – to enable selection of a local store pickup during the checkout process
* Add a new category of products, eg ‘Garden’ with an appropriate product inventory
* Investigate support for adding other products to the cart. Currently the ‘Flared Accent Chair’ is always added to the cart regardless which product button is pressed
* Implement the ‘Wish List’ functionality
* Simulate 30 products in the shopping cart – with a view to showing a UI overflow issue!
* Inject some ‘bad data’ which causes UI display issues
* Implement the search – for just a few static strings like Bed, Chair, Light etc