**Using Rendering Parameters to Inject Data into Hidden Input – Sitecore**

What are rendering parameters? They are the additional key-value pair data you can feed to the [Sitecore](https://www.sitecore.com/) component in addition to data source. The textbook example would be adding an additional CSS class to the component. Content authors can customise the style of the component, like background colour or button style, which in turn makes the component more reusable.

You can set rendering parameters either on the standard values of the template or content items or in Experience editor while editing the content.

The keys of rendering parameters can be defined by creating a parameter template. Alternatively, it can freewheel as you can type the key value pair in the Parameters section in Control Properties of the component.

The cool thing about rendering parameters is that you can mix both ways. No matter how many parameters you add, they are still stored in database as a query string and accessed by Sitecore API, RenderingContext.Current.Rendering.Parameters. The custom RenderingParameters object is like a dictionary. You can loop through by foreach and accessed by .Key and .Value.

By leveraging the nature of rendering parameters, let’s do something experimental. What if we create a component that takes all the parameters as a dictionary object and inject it into a hidden input as custom data. In this way, JavaScript can read these custom parameters to do unique behaviours depending on what is defined as parameters.

Can we create a reusable component to inject a hidden input with custom data? Yes, we can and that is exactly what we are going to build.

This is the component output.

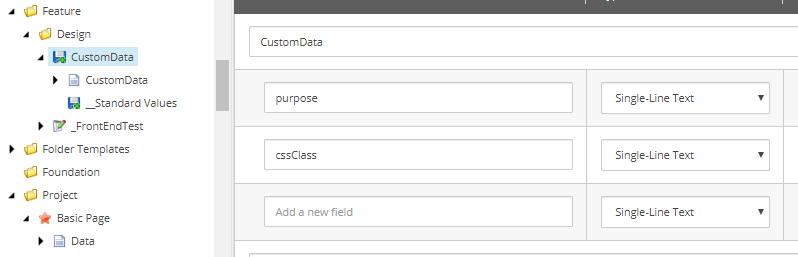
<script src="https://gist.github.com/mydatahack/8eb4125f8f96a0bef69c727712ead2c5.js"></script>

**Steps**

We need to create templates, controller rendering and add parameters. We will get to the coding part later.

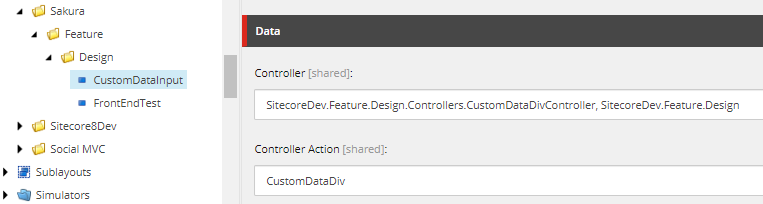
1. **Create parameter template**

First, let’s create parameter template. Make sure to inherit from Templates/System/Layout/Rendering Parameters/Standard Rendering Parameters.

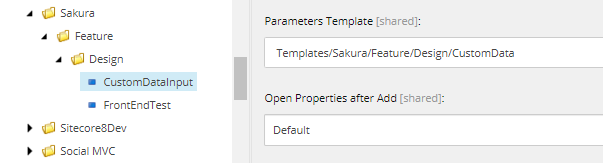


1. **Create Controller Rendering**

Let’s reference to the correct controller & action.

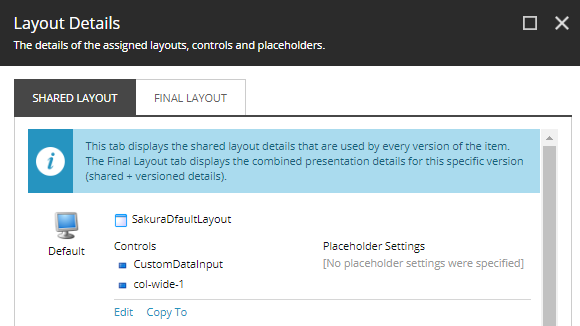


Then, scroll down to find the Parameter Template section and add the path to the template we created in step 1.



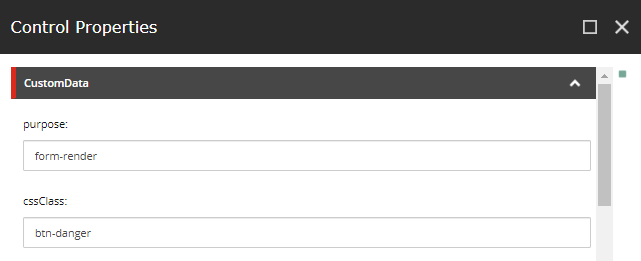
1. **Add Controller Rendering to the page template**

Now, we can add the component created above to the page template.

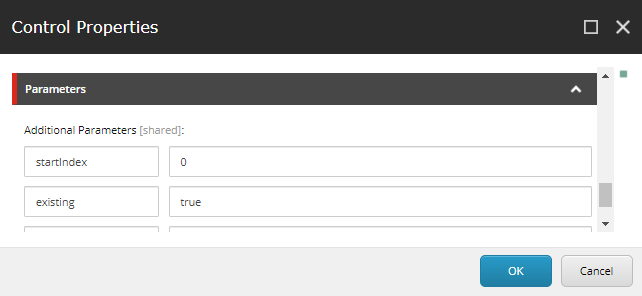


1. **Add parameters in Control Properties of the component**

You will see it at the top of Control Properties of the component.



Alternatively, it can be defined as typing the key value pair in the Parameters section in Control Properties of the component. Let’s add some.



Now, it’s time to code.

1. **Code!**

Coding is fairly straight forwards. But, it is fun. Check this out.

**Model**

This is a simple model with one dictionary field that has all the parameters stored as dictionary.

<script src="https://gist.github.com/mydatahack/35fe152391a2a091353296dfcc57b503.js"></script>

**View**

It takes the dictionary and serialise it as Json. Then, inject it into custom-data attribute in the hidden input.

<script src="https://gist.github.com/mydatahack/d011a99e6446b6e2bb00dca1b12a52a5.js"></script>

**Controller**

Using Sitecore API to get parameters, passing it to a repository method to create a model and returning the view with the model.

<script src="https://gist.github.com/mydatahack/2c97e1ffb505a3d48929693729ffa009.js"></script>

**Repository**

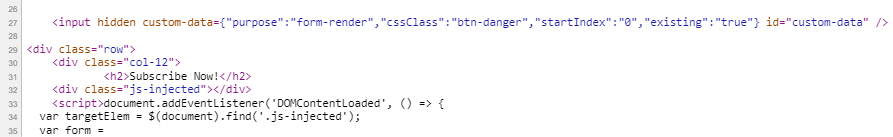
Takes parameters and converting it into a dictionary object.

<script src="https://gist.github.com/mydatahack/fa7d665605e4fdcf77cb1feb76aa39e1.js"></script>

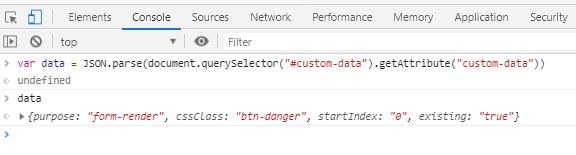
That’s it!

1. **Test the component**

You can check the source to see if it exists.



See if you can access the data by JavaScript.



It works!

Yay!