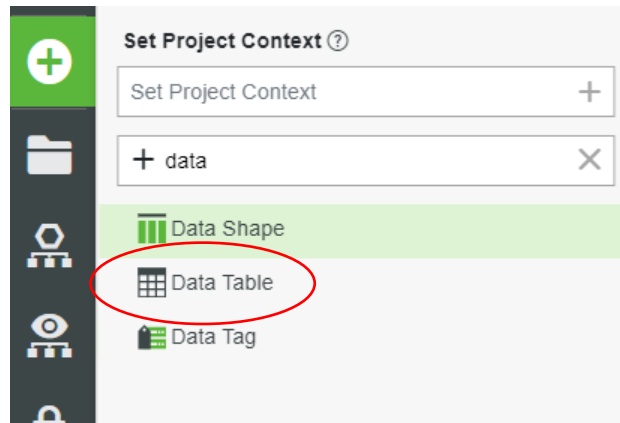










Displaying Vuforia Studio data in a Thingworx Mashup

1. Open Thingworx and create a new DataTable → DataTable




- a. Name = FirstInitials_TestDataTable (ex. JS_TestDataTable)
- b. Add a Project and a DataShape. (In this example I use a previously made project: Test_DatatableToMashup and DataShape: Test_BikeDataShape). You can create your own Project or use PTCDefaultProject. Your DataShape should have 4 field definitions:
 - i. An Image (name it InitialsBikeImage: ex. JSBikeImage)
 - ii. A String named InitialsNameInput: ex. JSNameInput.
 - iii. An Integer named InitialsImageNumber: ex. JSImageNumber
 - iv. A String named InitialsImageText: ex. JSImageText

Order	Name	Actions	Additional Info	Default Value
<input type="checkbox"/>	 JSBikeImage			
<input type="checkbox"/>	 JSNameInput			
<input type="checkbox"/>	123 JSImageNumber			
<input type="checkbox"/>	 JSImageText			

Data Table: New Data Table - 29 * ? To Do ▼ C

General Information Properties and Alerts Services

 **Name** ? (required)
JS_TestDataTable
[Change](#)

Description ?

Project ? (required)
Test_DatatableToMashup ✕
[Set as project context](#) ?

Tags ?
 +

Base Thing Template ? (required)
DataTable

Implemented Shapes ?
 +

Value Stream ?
 +

Data Shape ? (required)
Test_BikeDataShape ✕

Persistence Provider ? (required)

2. Go to Properties and Alerts, add 4 properties
 - a. Name = BikeImage → BaseType = Image → check box Persistent
 - b. Name = ImageText → BaseType = String → check box Persistent
 - c. Name = NameInput → BaseType = String → check box Persistent

d. Name = PictureNumber → BaseType = integer → check box Persistent

New Property 30

Name (required)
BikeImage

Description

Base Type
IMAGE

Has Default Value

☒ Persistent

☐ Read Only

☐ Logged

Binding
None

[Advanced Settings](#)

Properties | Alerts

My Properties

Name	Actions	Source	Default Value	Value	Alerts	Category	Additional Info
BikeImage					0		
ImageText					0		
NameInput					0		
PictureNumber					0		

3. Go to Services, press ADD → Local JavaScript

a. Name = AddImage

b. Under the Inputs tab, click +Add

i. Name = newImage → BaseType = Image

ii. Name = newUserName → BaseType = String

General Information | Properties and Alerts | **Services**

Services

New Service Local (JavaScript) ☒

Service Info

Name (required)
addImage

Description

Category

1

NEW SERVICE Local (JavaScript)

Service Info

Inputs

+ Add

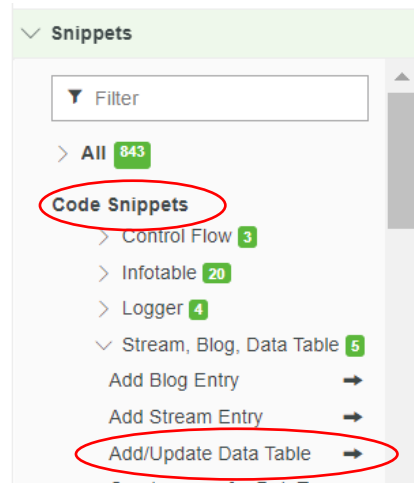
newImage

newUserName

Output

Snippets

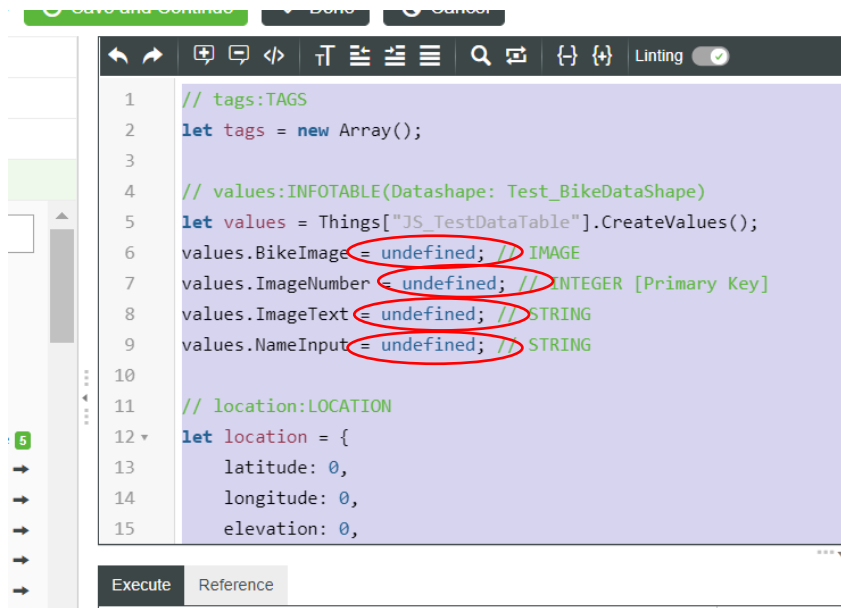
- c. Under Snippets tab, expand Stream, Blog, Data Table
 - i. Click the arrow “→” next to “Add/Update Data Table”



- ii. Pick the DataTable you just made. This will populate your text area.
- d. On lines 4-9 it should give you code similar to this:

```
// values:INFOTABLE(Datashape: Test_BikeDataShape)
let values = Things["JS_TestDataTable"].CreateValues();
values.BikeImage = undefined; // IMAGE
values.ImageNumber = undefined; // INTEGER [Primary Key]
values.ImageText = undefined; // STRING
```

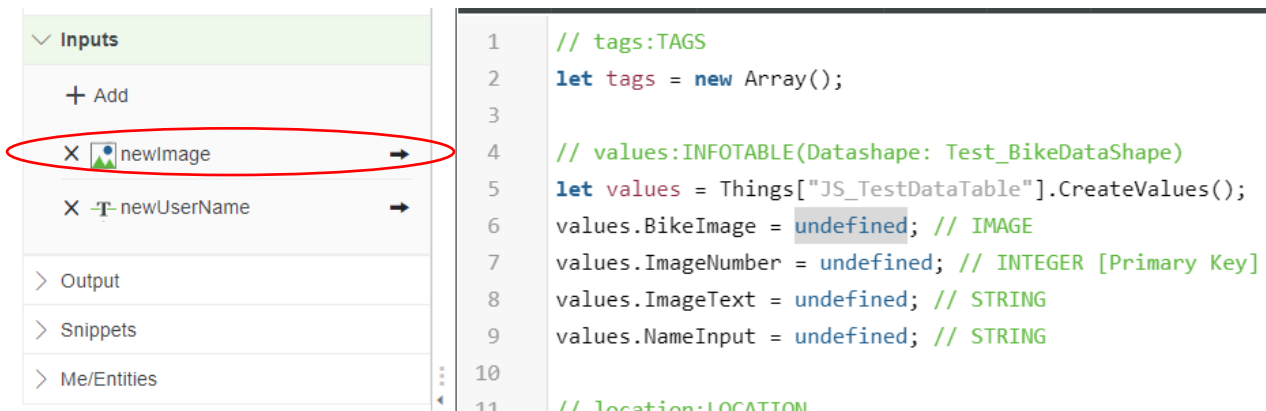
```
values.NameInput = undefined; // STRING
```



```
1 // tags:TAGS
2 let tags = new Array();
3
4 // values:INFOTABLE(Datashape: Test_BikeDataShape)
5 let values = Things["JS_TestDataTable"].CreateValues();
6 values.BikeImage = undefined; // IMAGE
7 values.ImageNumber = undefined; // INTEGER [Primary Key]
8 values.ImageText = undefined; // STRING
9 values.NameInput = undefined; // STRING
10
11 // location:LOCATION
12 let location = {
13   latitude: 0,
14   longitude: 0,
15   elevation: 0,
```

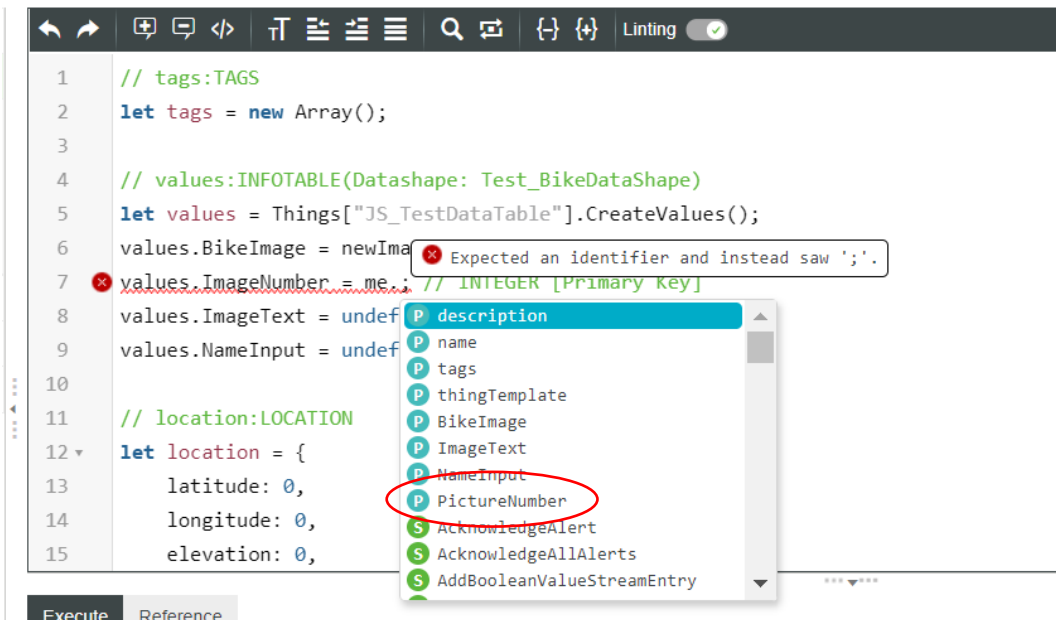
e. Repopulate all the “undefined” values:

- i. Double click on undefined on line 6. While “undefined” is highlighted, go over to the inputs on the left side of the screen and push the “→” next to newImage. This should repopulate “undefined” with “newImage”



```
1 // tags:TAGS
2 let tags = new Array();
3
4 // values:INFOTABLE(Datashape: Test_BikeDataShape)
5 let values = Things["JS_TestDataTable"].CreateValues();
6 values.BikeImage = undefined; // IMAGE
7 values.ImageNumber = undefined; // INTEGER [Primary Key]
8 values.ImageText = undefined; // STRING
9 values.NameInput = undefined; // STRING
10
11 // location:LOCATION
```

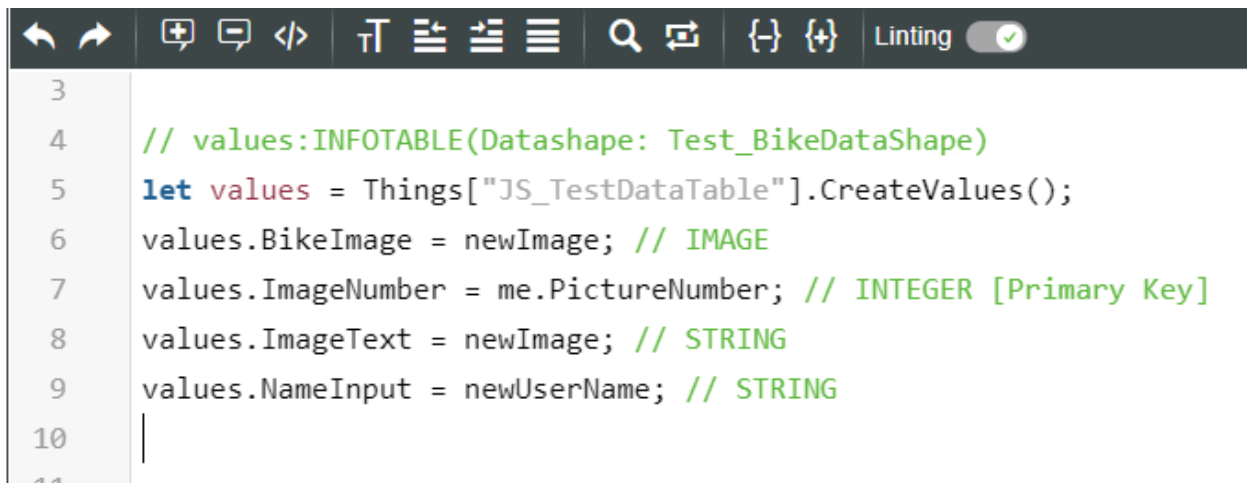
- ii. Double click on undefined on line 7. Type “me.” This should give you options of properties to pick. Choose “PictureNumber”. End the line with a “;”



- iii. Double click on undefined on line 8. Repeat Line 6 and repopulate “undefined” with input “newImage”
- iv. Double click on undefined on line 9. Using the newUserName input you made, replace “undefined” on line 9. This should repopulate “undefined” with “newUserName”
- v. Your code should now match this:
- ```

// values:INFOTABLE(Datashape:
Test_BikeDataShape)
let values =
Things["JS_TestDataTable"].CreateValues();
values.BikeImage = newImage; // IMAGE
values.ImageNumber = me.PictureNumber; //
INTEGER [Primary Key]
values.ImageText = newImage; // STRING
values.NameInput = newUserName; // STRING

```



```

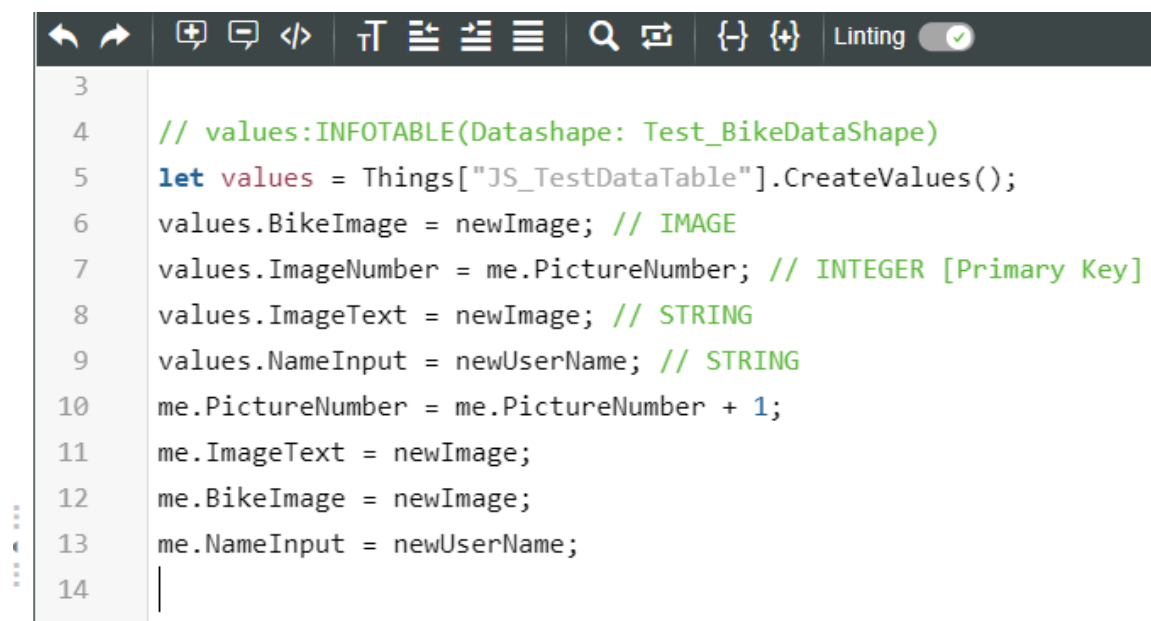
3
4 // values:INFOTABLE(Datashape: Test_BikeDataShape)
5 let values = Things["JS_TestDataTable"].CreateValues();
6 values.BikeImage = newImage; // IMAGE
7 values.ImageNumber = me.PictureNumber; // INTEGER [Primary Key]
8 values.ImageText = newImage; // STRING
9 values.NameInput = newUserName; // STRING
10
11

```

- f. Copy and paste this code after line 9
- ```

me.PictureNumber = me.PictureNumber + 1;
me.ImageText = newImage;
me.BikeImage = newImage;
me.NameInput = newUserName;

```



```

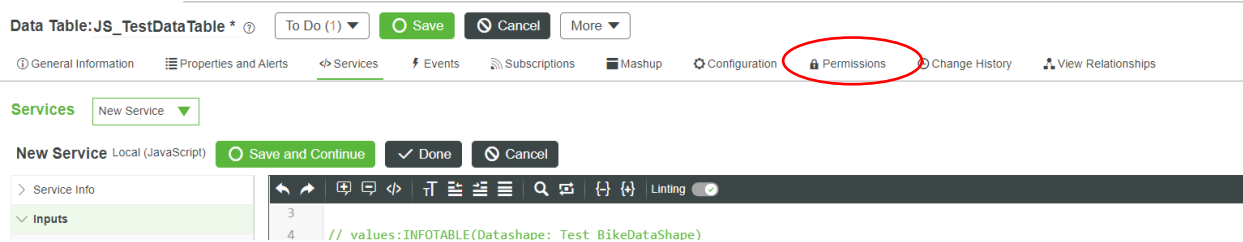
3
4 // values:INFOTABLE(Datashape: Test_BikeDataShape)
5 let values = Things["JS_TestDataTable"].CreateValues();
6 values.BikeImage = newImage; // IMAGE
7 values.ImageNumber = me.PictureNumber; // INTEGER [Primary Key]
8 values.ImageText = newImage; // STRING
9 values.NameInput = newUserName; // STRING
10 me.PictureNumber = me.PictureNumber + 1;
11 me.ImageText = newImage;
12 me.BikeImage = newImage;
13 me.NameInput = newUserName;
14
15

```

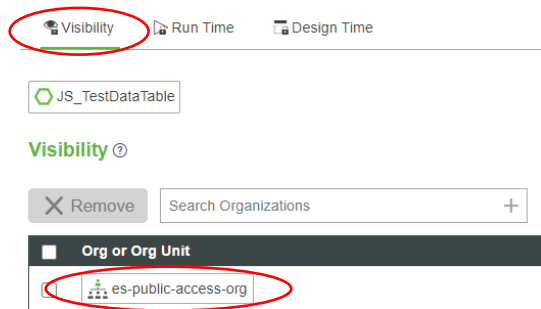
- g. Click “Save and Continue” and then “Done”

4. Change Permissions in Data Table

a. Go to Permission tab



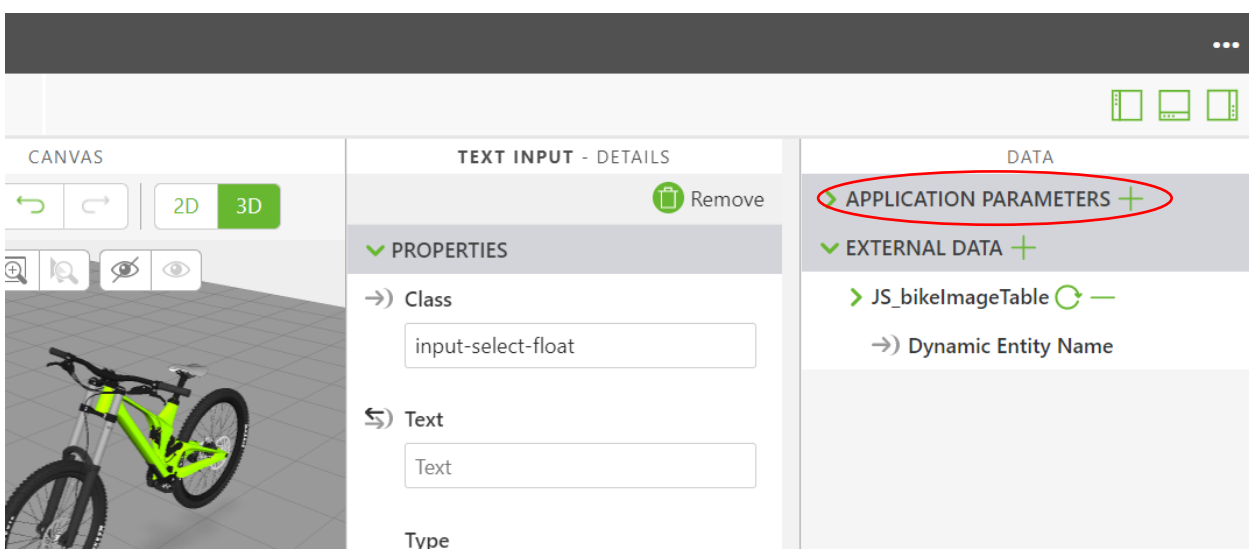
b. Under Visibility, search and select “es-public-access-org”



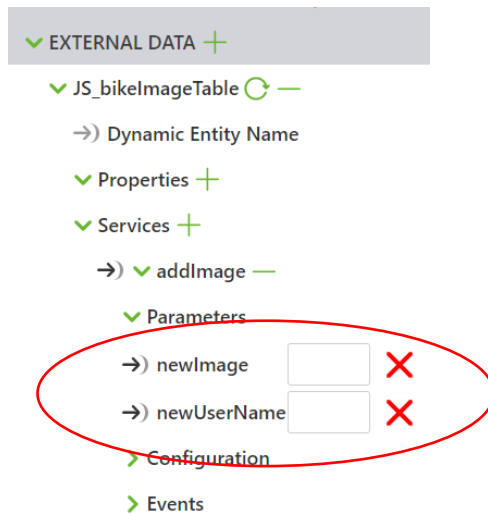
c. Under Run Time, search and select “es-public-access-org” and then check each box

5. Open Vuforia Studio → Bike Example

a. On the far right column under external data, click on the + on “external data +”

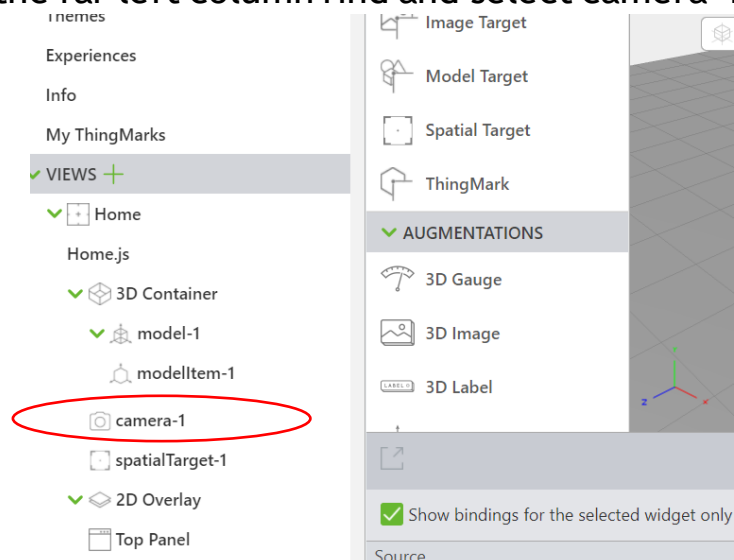


- i. On the left tab of the popup, type in your initials and select your DataTable
- ii. On the right tab of the popup, click the middle tab “Services”
- iii. Search + for “addImage”
- b. Under External Data, extent your Data Table → Services → addImage → Parameters, until you see NewImage and NewUserName

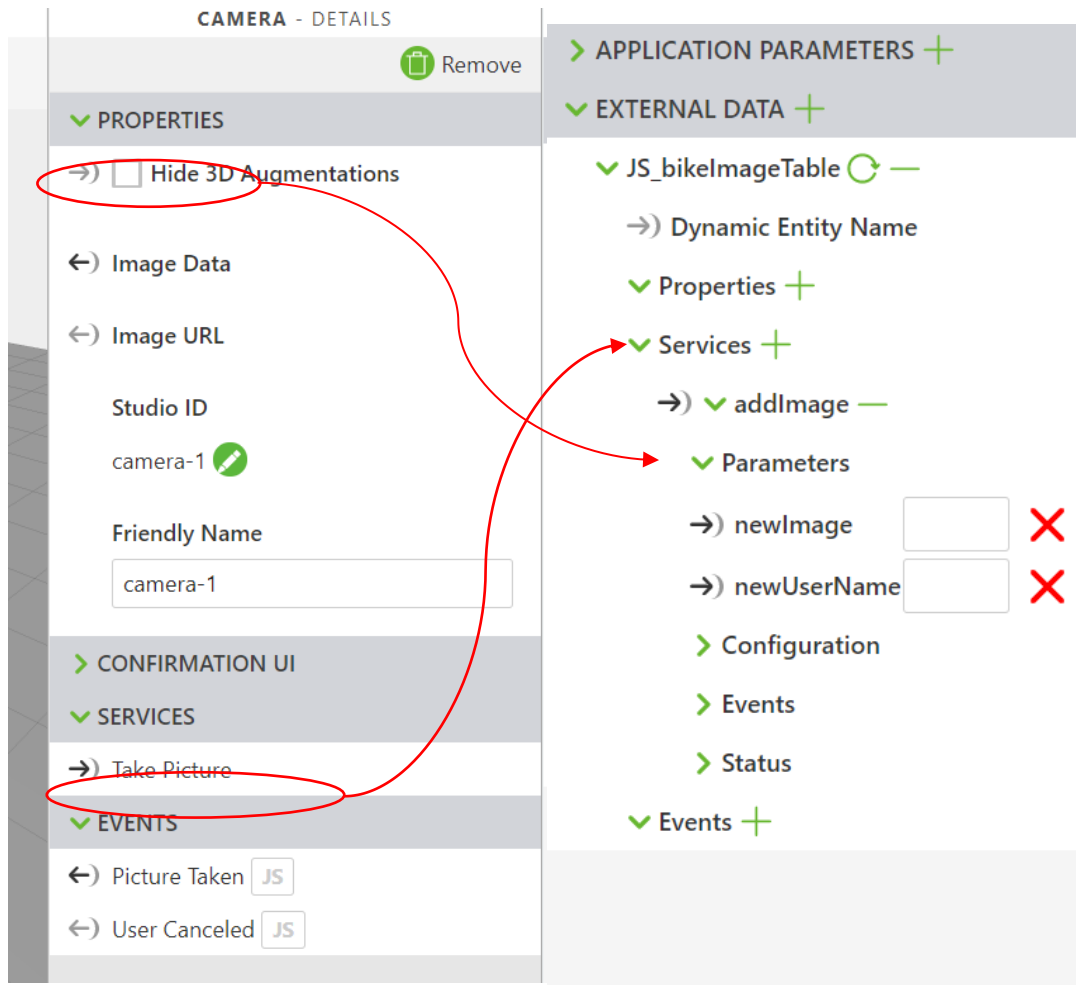


6. Bind Data to Thingworx:

- a. On the far left column find and select camera-1

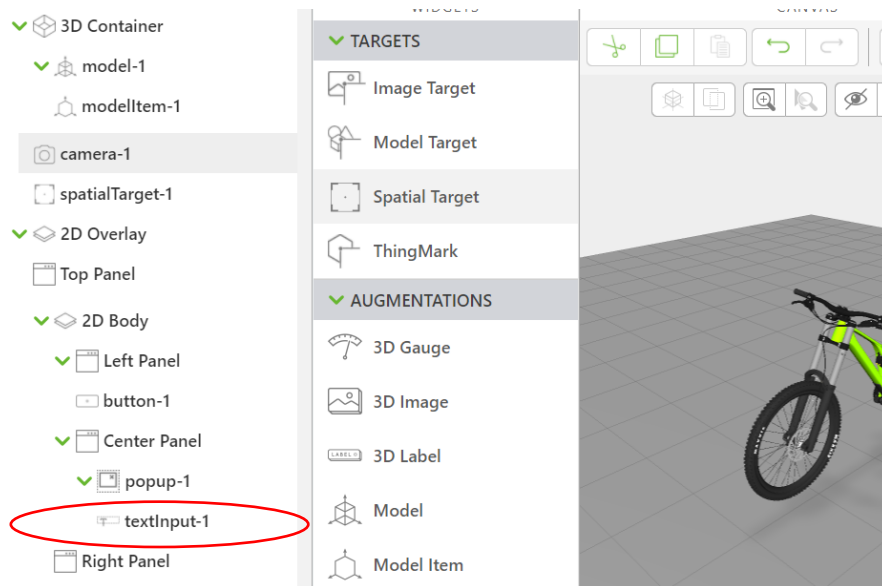


- b. On the middle right panel, find “Picture Take” under Events, click and drop the “←” Picture Taken to “addImage” on the far right column

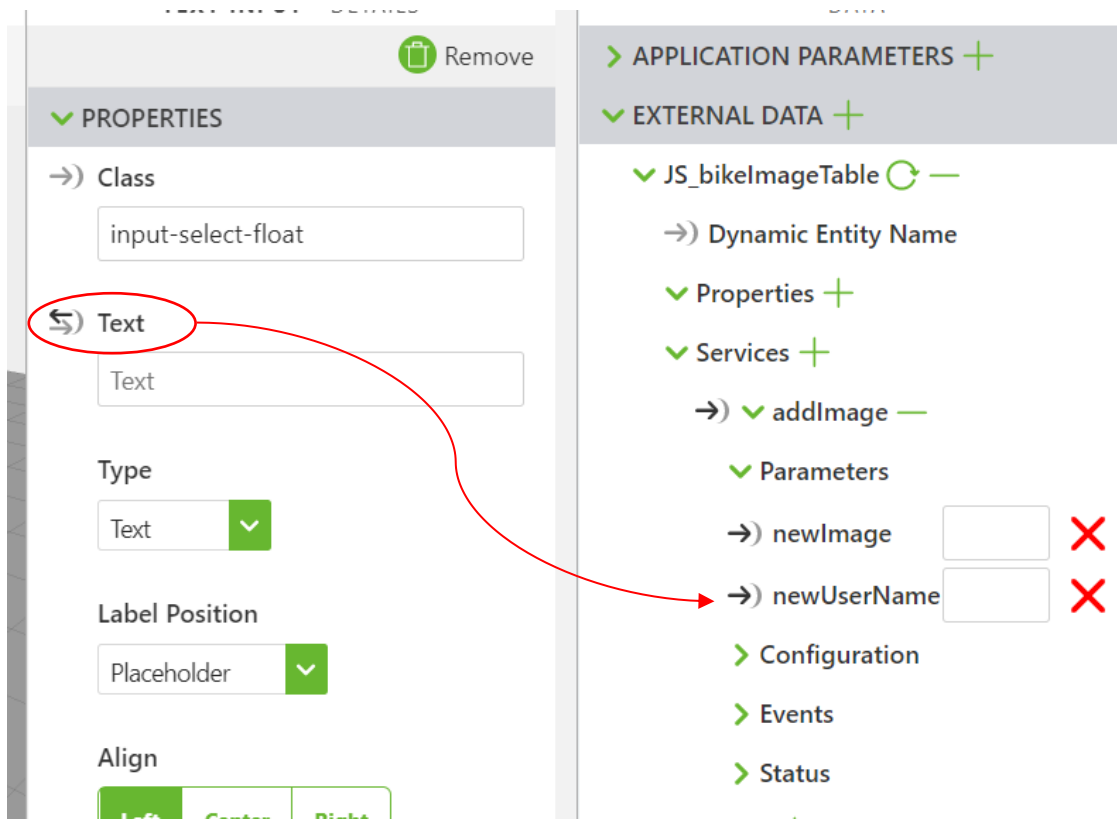


- c. On the same middle-right panel, drag and drop the “←” from Image Data to “New Image” on the far right column

d. On the far left column, find and select “textInput-1”

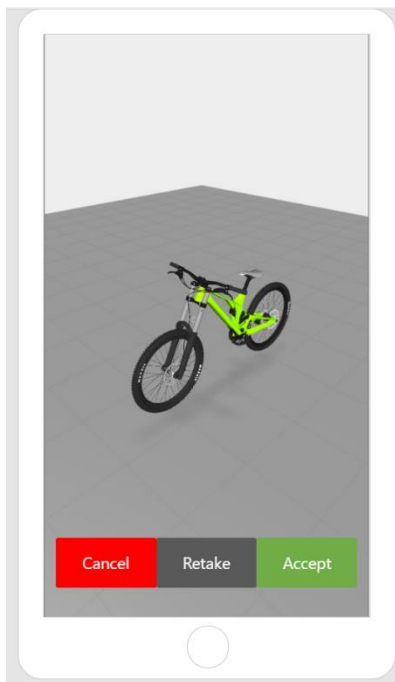


e. Click and drag the arrow from “text” to “newUserName” on the far right column






















- f. Click Save
7. Click Preview
 - a. Type you Name and then press enter.
 - b. Select the camera button.

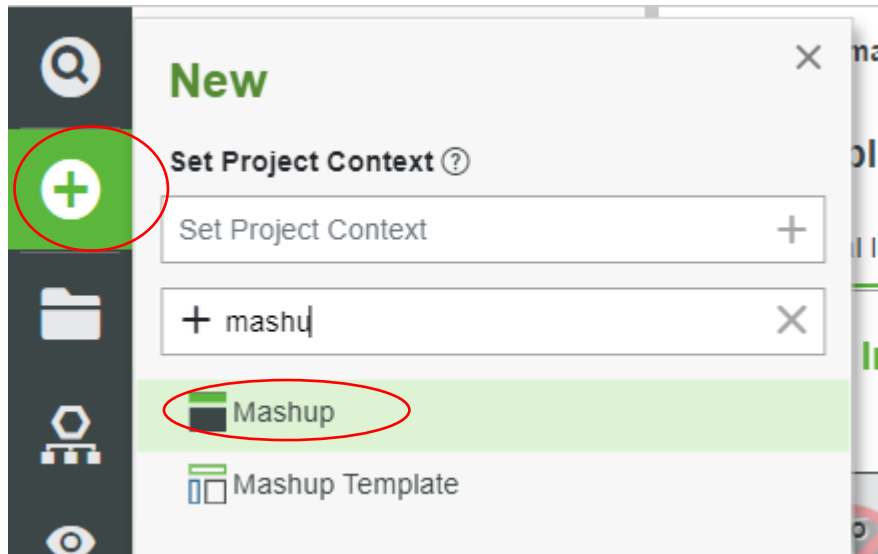
c. Press Accept



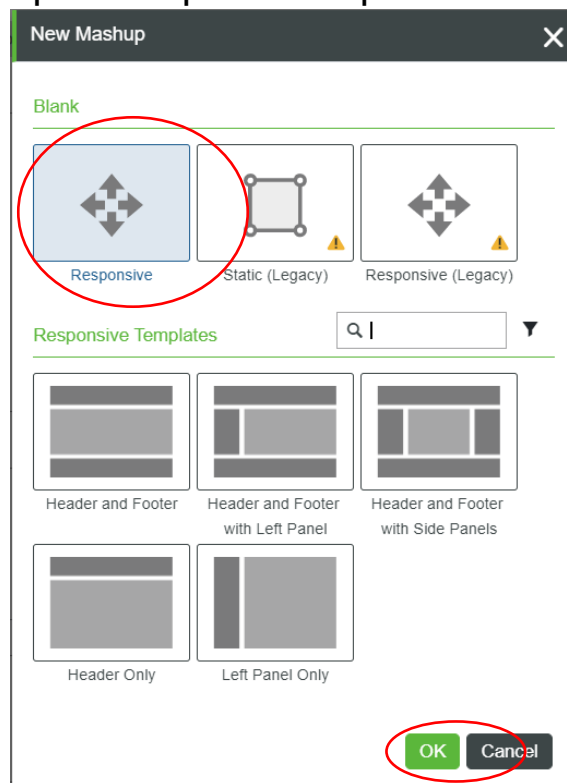
8. Go back into Thingworx to your Data Table and press refresh. In the Properties tab you should see the image you just took in Vuforia along with your name

My Properties + Add Duplicate Delete Manage Bindings Refresh									
Name	Actions	Source	Default Value	Value	Alerts	Category	Additional Info		
<input type="checkbox"/>  bikeImage					 0				
<input type="checkbox"/>  imageText				 /9j/4AAQSkZJRgABAQAAQABA...	 0				
<input type="checkbox"/>  nameInput				 Jessica Stasny	 0				
<input type="checkbox"/> 123 pictureNumber				 15	 0				

9. Create a new Mashup

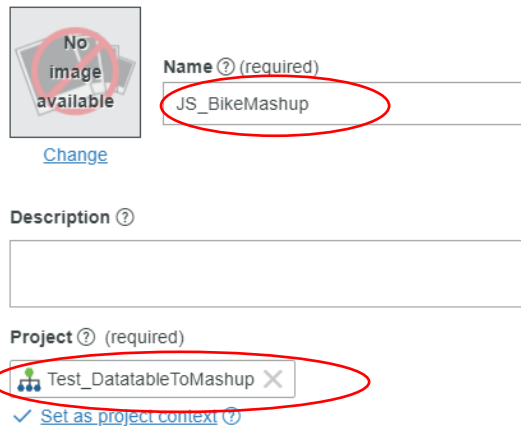


10. In the popup screen press “Responsive” and OK



11. Name your Mashup: FirstName_BikeMashup (Ex. JS_BikeMashup)
- a. For Project, search and select Test_DatatableToMashup

General Information



Name ? (required)
JS_BikeMashup

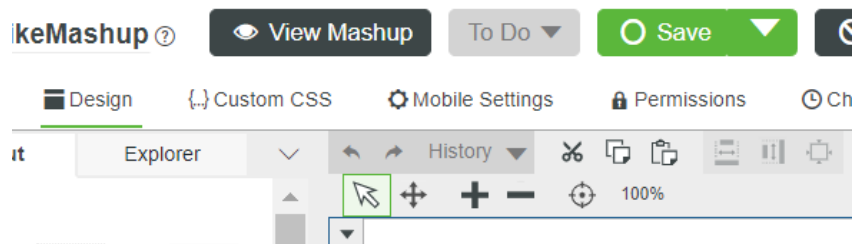
Description ?

Project ? (required)
Test_DatatableToMashup

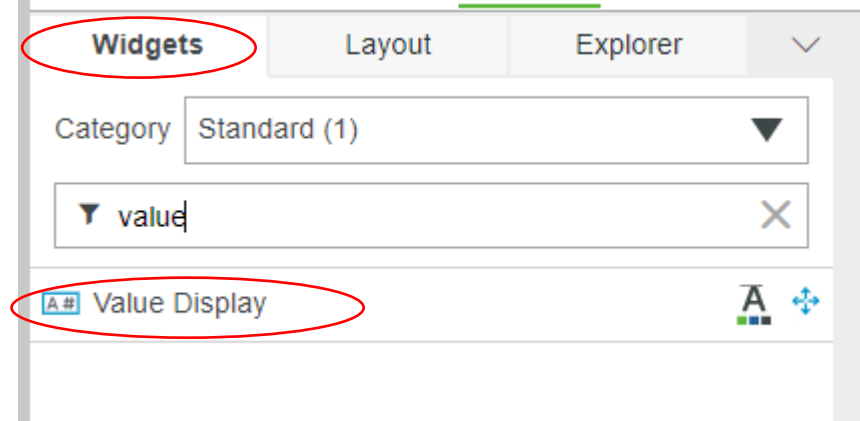
[Change](#)

[Set as project context](#)

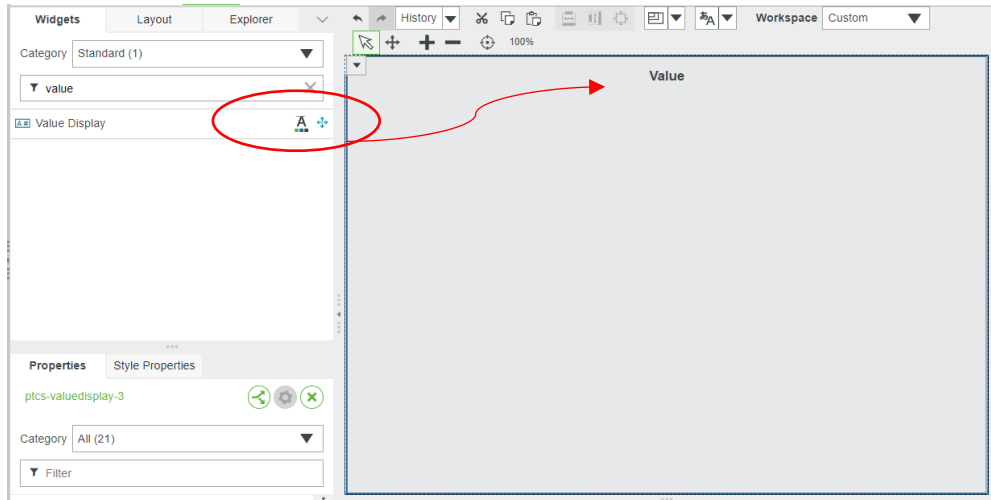
b. Press Save and go to the Design tab



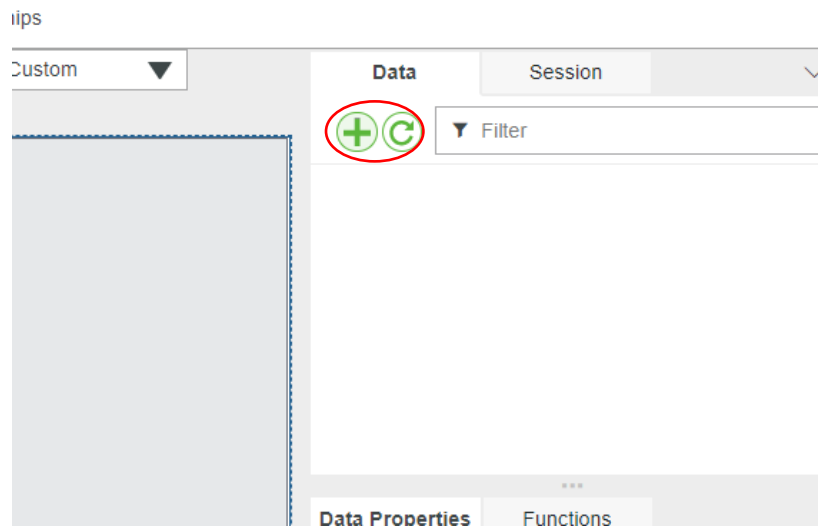
c. Under the “widgets” tab search for value display



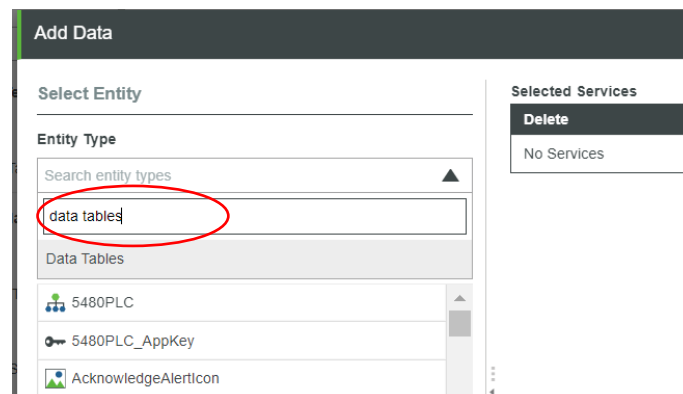
d. Click and drag the Value Display into the Mashup Panel in the center of the screen



12. In the top right panel select the “+” button under the data tab



- a. In the popup under Entities, search for Data Tables



- b. Find and select your Data Table

Add Data

Select Entity / Select Service(s)

Selected Entity

JS_TestBikeDataTable X

☐ Show dynamic services ?

Select Service Category

Choose category ▼

c. Under Services, search and select for “getProperties”.

Select Entity / Select Service(s)

Selected Entity

JS_TestBikeDataTable X

☐ Show dynamic services ?

Select Service Category

Choose category ▼

Services

getproperties X

+ GetProperties →

d. Press the “→” arrow next to “getProperties” to add it and check “Execute on Load”

e. Click “Done”

Add Data

Select Entity / Select Service(s)

Selected Entity

JS_TestBikeDataTable X

☐ Show dynamic services ?

Select Service Category

Choose category ▼

Services

getproperties X

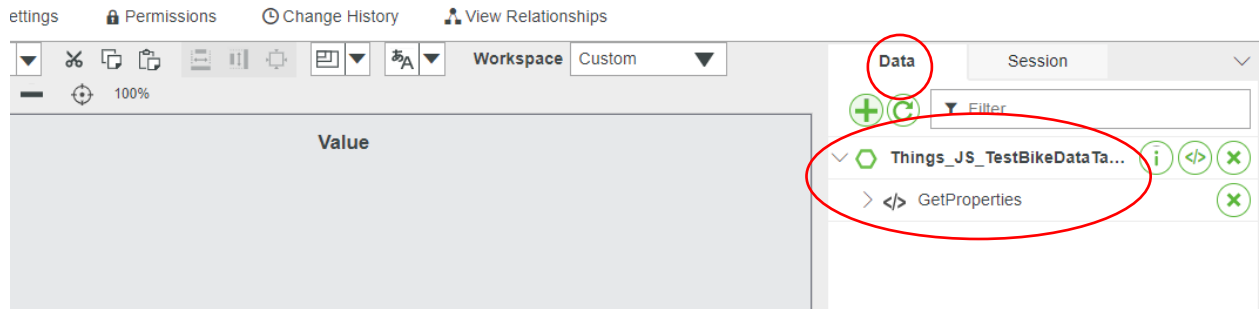
+ GetProperties →

Selected Services

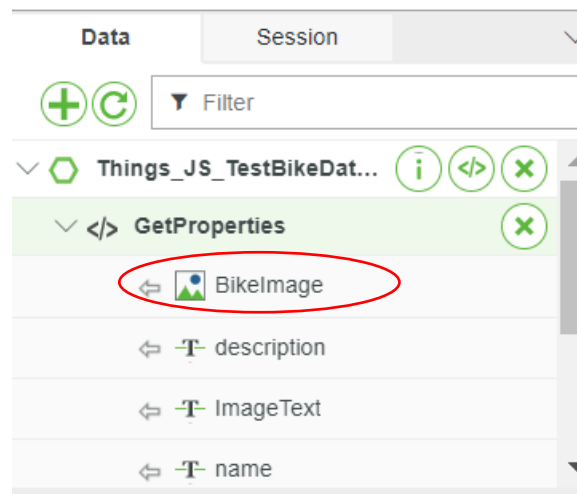
Delete	Entity	Service	Execute on Load
X	JS_TestBikeDataTable	GetProperties	<input checked="" type="checkbox"/>

Done **Cancel**

f. You should now be able to see “getProperties” under the Data tab as shown below

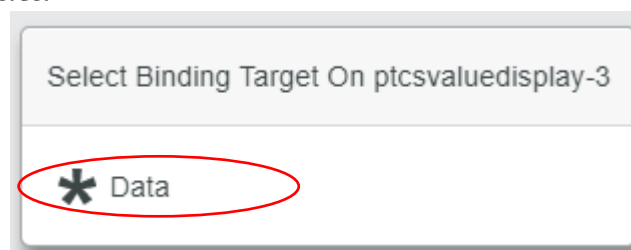


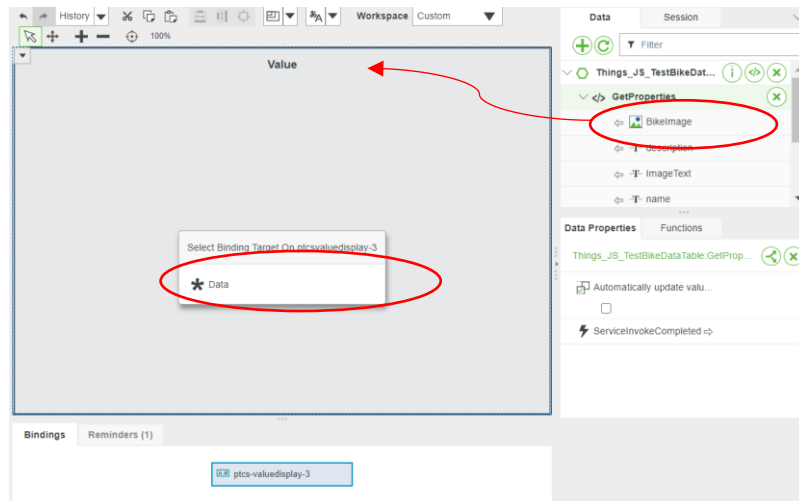
g. Expand the GetProperties tab until you see “BikeImage”



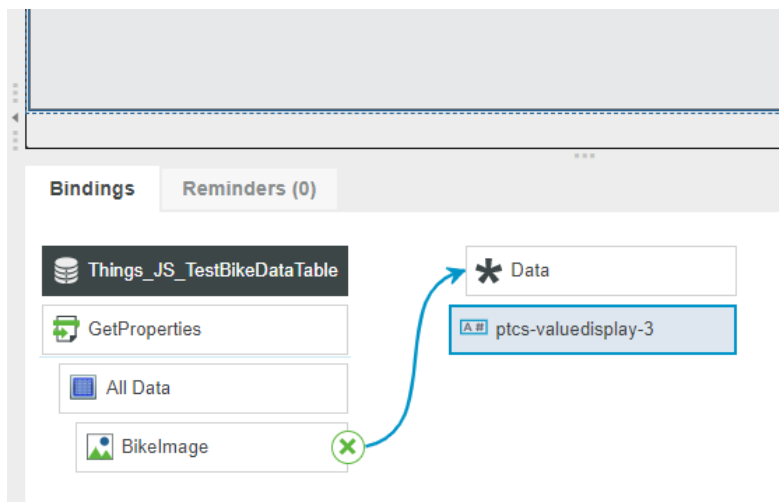
h. Click and drag the “←” button next to BikeImage into the Value Display in the center of the screen. A little popup should appear on the mashup screen.

i. Select “*Data”

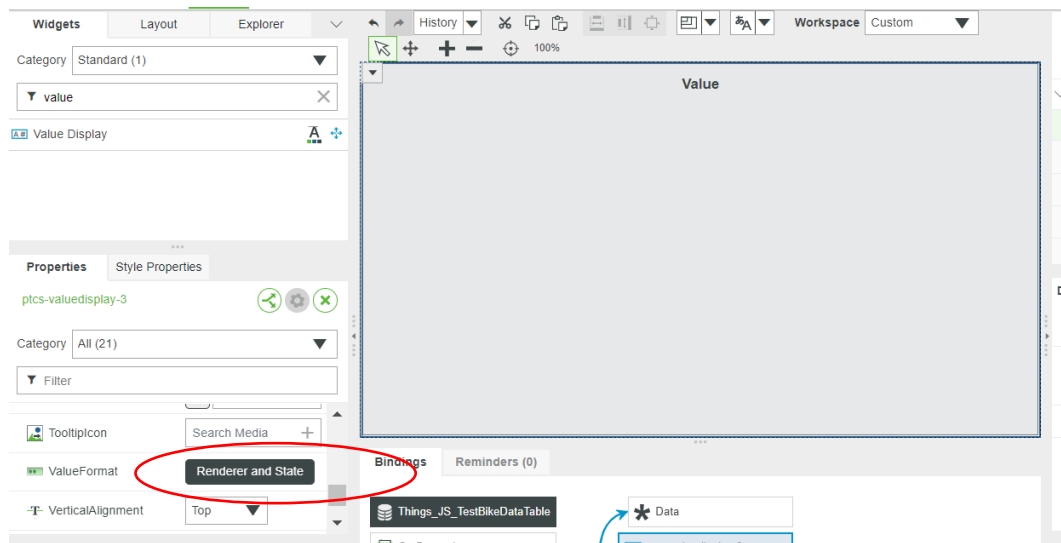




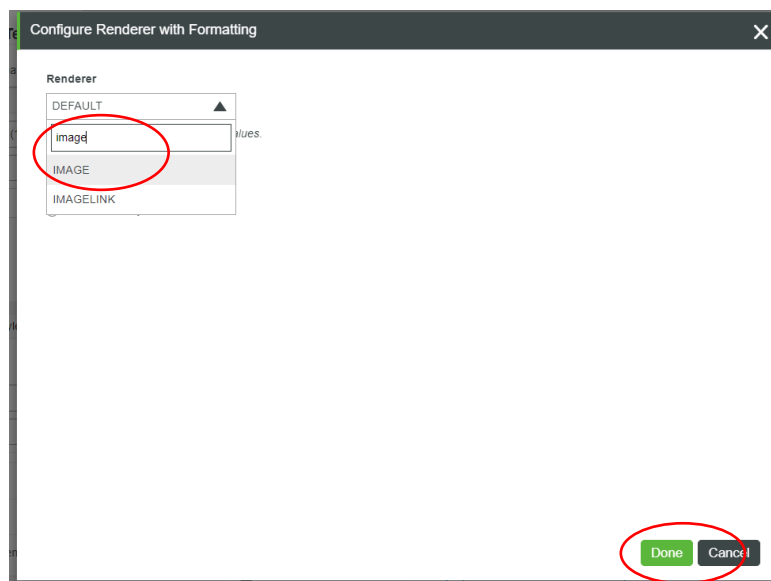
- j. At the bottom of the screen under bindings you should see the BikeImage property now bound to the valuedisplay widget.



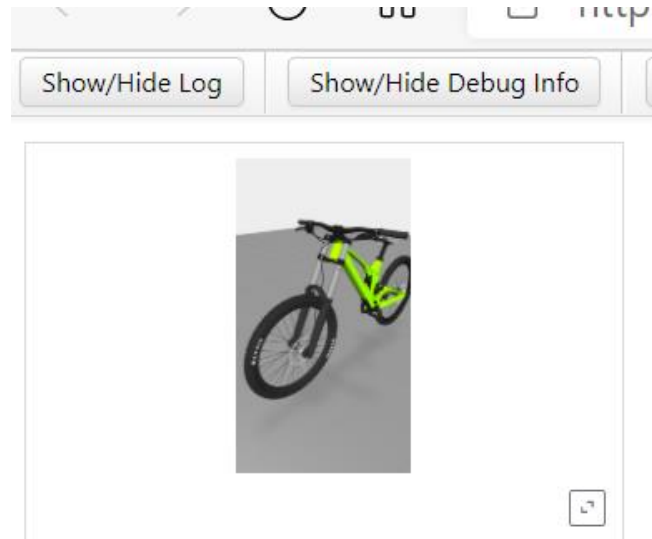
- k. Click on the center of the screen on the Value widget in the mashup. Under the Properties tab on the bottom left panel of the screen, scroll down until you see “Value Format”



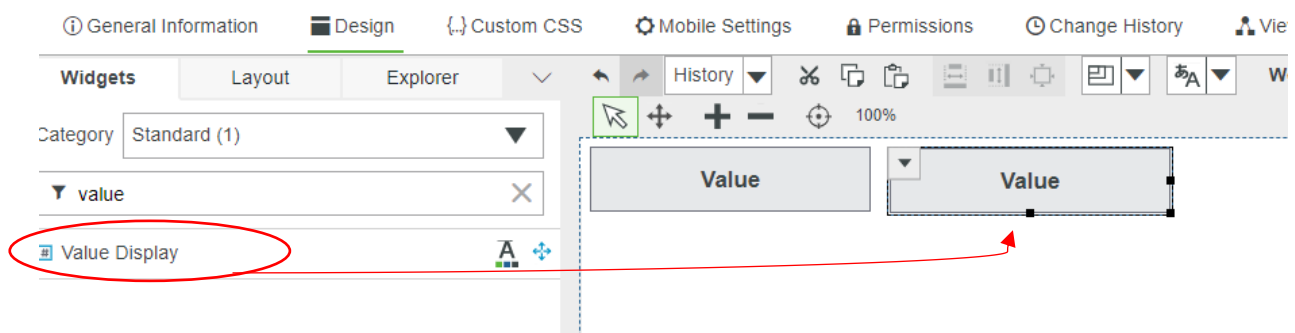
- l. Select “Render and State”
- m. Under Renderer, search and select “Image” and then press “Done”.



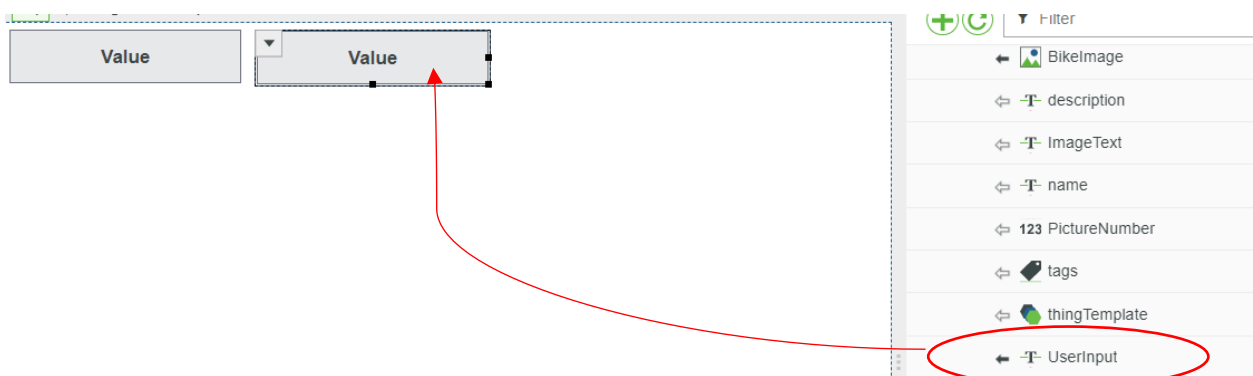
- n. Press “Save” and “View Mashup”. An image of the most recent picture taken should appear on the screen.



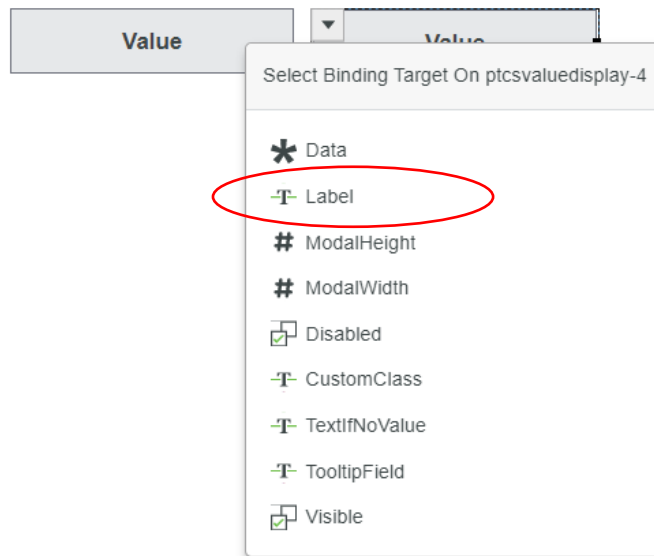
13. To display the username, insert another value display on the screen by dragging and dropping the widget into the center screen again.



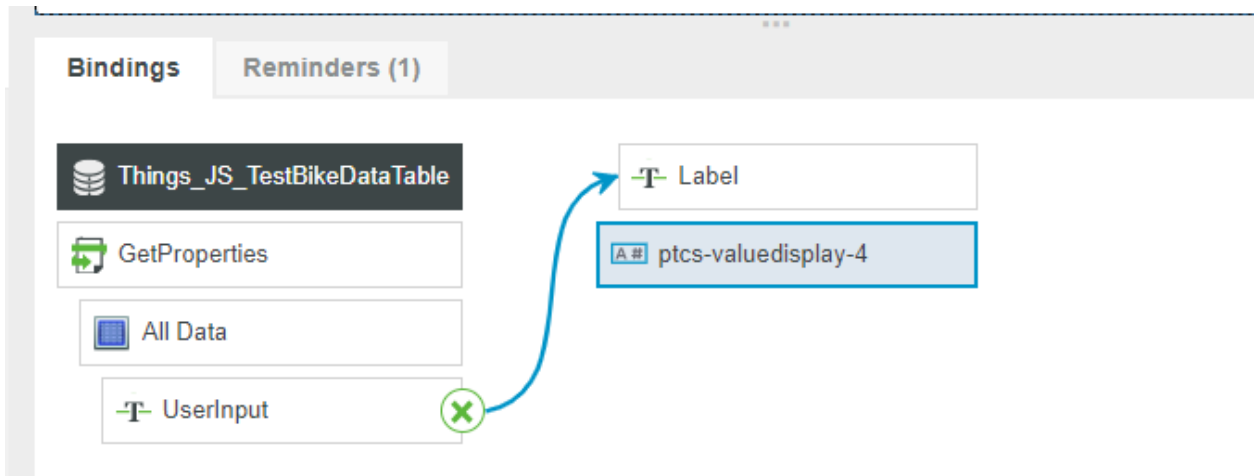
- a. On the far right panel, select and drag the “←” button next to “UserInput” into the second and newest value display widget in the middle of the screen



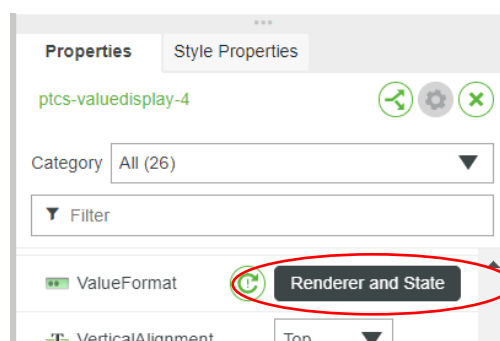
- b. In the popup, select “label”

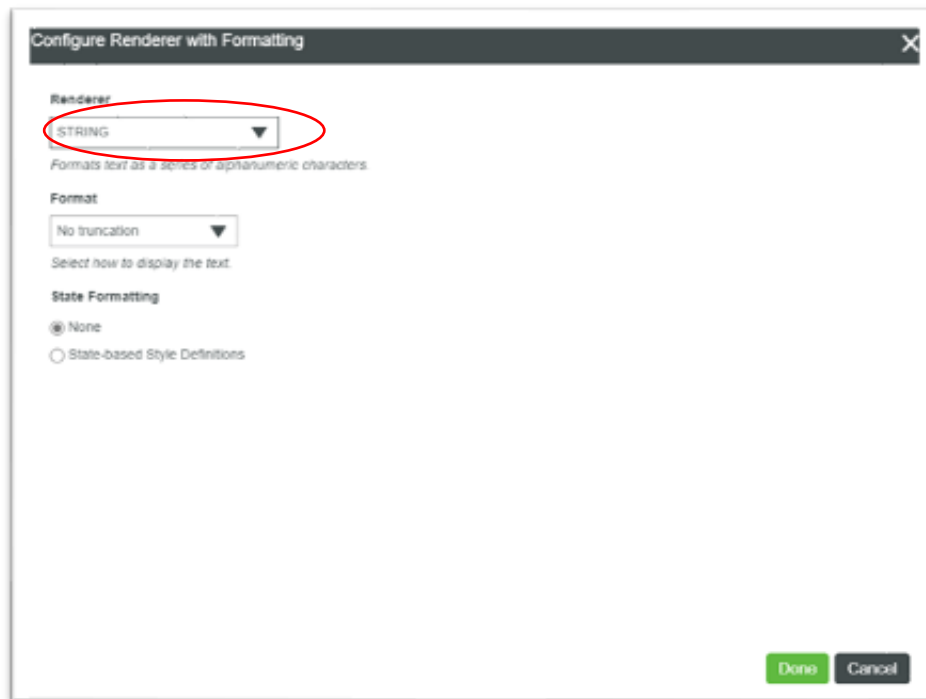


- c. At the bottom of the screen you should now see the userInput bound to the second value display widget



- d. In the bottom left panel select "Renderer and State" and change the Renderer from "default" to "String"





- e. Select “Save” and “View Mashup”. You should now see the most recent Image and the user who took the picture.

