PTC®

PTC® IoT Academic Program

Predictive Weather Mashup

How to build a Predictive Weather Mashup – Phase One

Revision #	Date	ThingWorx Revision	Changes	Owner
1.0	**03.11.2014**	**5.0.2.33**		**Adrian Petrescu**
2.0	**02.12.2014**	**5.0.2.33**		**Adrian Petrescu**
3.0	**22.12.2014**	**5.0.2.33**		**Adrian Petrescu**
4.0	**14.07.2015**	**5.0.2.33**	**Minor changes**	**Veronica MIHAI**
5.0 – Current	**20.10.2015**	**5.0.2.33**	**Changed OpenWeatherURL**	**Veronica MIHAI**

Acknowledgements

We would like to thank Christopher Wasden, EdD, Professor of Innovation, Executive Director, Sorenson Center for Discovery and Innovation and Associate Executive Director, Center for Medical Innovation for his input into the design of the this application. Professors Wasden commitment to this project reflects an understanding that thinking about that data that you need to collect and how you will use the data in critical to designing smart connected products that will bring value to the people who will use those products.

PREDICTIVE WEATHER MASHUP PHASE ONE

Predictive Weather Mashup (Phase 1)

Overview

This document shows you how to build a simple weather mashup using the ThingWorx Composer™ development tool and then how to access it using any common browser or tablet.

If you have already viewed the Simple Weather Mashup video and pdf, the first steps of this documentation may be redundant. If this is your first interaction with the ThingWorx platform you will need to read and learn about creating an application also known as mashup.

This use case details a predictive weather mashup that will obtain weather data information from the Yahoo Weather web site. Our predictive weather mashup obtains the following weather data hourly: weather condition, temperature, humidity, wind speed, wind direction, precipitation, sunrise and sunset for a certain city. The information is displayed in a mashup.

There will be 5 simple steps covered during the creation of this mashup. This will then be expanded through subsequent phases. The following steps outline how the simple mashup is built:

- Log in to the ThingWorx application and create a Model Tag;
- Create a thing called “WeatherThing”;
- Create a service for the “WeatherThing”;
- Create a mashup called “WeatherMashup”;
- Apply GetYahooWeatherInformation service to the WeatherMashup and bind the data;

Step 1: Log in to the ThingWorx application and create a Model Tag

In this step, you will create a Model Tag that you will use to identify all your mashup components.

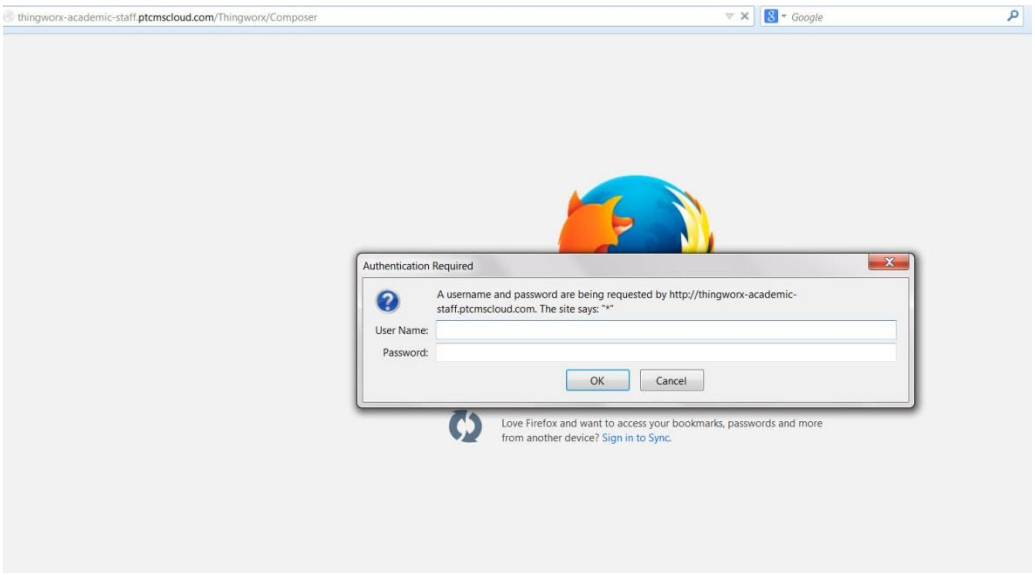
*ThingWorx **model tags** are a mechanism to label ThingWorx objects and data to assist in grouping, filtering, and locating ThingWorx objects and searching/finding data efficiently. Model tags provide metadata about an object/entity in ThingWorx or a row/record recorded as data inside of a ThingWorx object/entity.*

A tag is a combination of a vocabulary and a specific vocabulary term. A tag is shown as Vocabulary: VocabularyTerm. Almost every ThingWorx entity can be tagged. Tags can be used to create a relationship between many different ThingWorx entities.
A vocabulary is a list of words or identifiers used to describe a particular concept or thing. A vocabulary can contain multiple vocabulary terms. An example of a vocabulary would be Colors or Locations.
A vocabulary term is a word or phrase used to describe a thing or to express a concept. Some examples of vocabulary terms of the Colors vocabulary are red, blue, and green. For Locations, it might be city names of plants.

ThingWorx has a naming restriction. When you set a name to a component in ThingWorx, you can't use space (blank) inside the name box, this means that all your component names will be in only one word (for example when you want to create the Temp And Humidity tag, you will type in name box "TempAndHumidity" in one word, and not "Temp And Humidity"). Also, there can't be 2 components with the same name in the same ThingWorx instance. Remember, this applies to everything in ThingWorx (not just Model Tags). This is important because each university shares the same instance

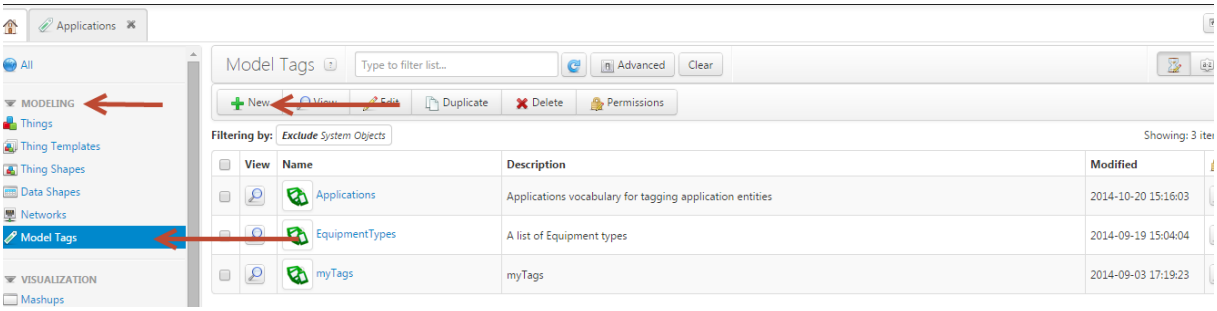
Log in to the ThingWorx Composer environment

- *Log in to the Composer application with the URL and the credentials you were supplied with*

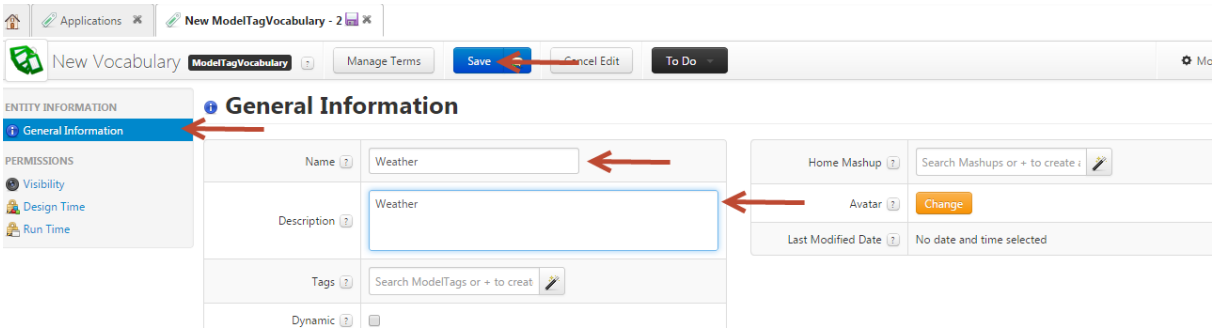


Create a Model Tag

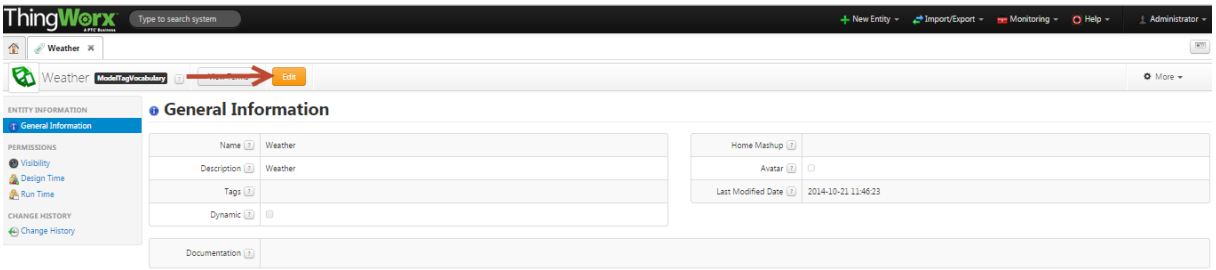
- *Select the **Model Tags** option from the **Modeling** section in the Home tab and click the **New** button.*



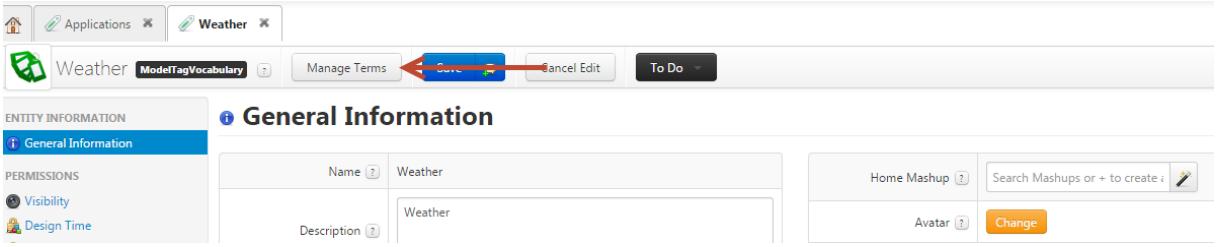
- *Type a Name and Description for the new Model Tag in the **General Information** section and save it (don't forget about the name restrictions and be sure your name is not the same with another students or teams name). An example would be "WeatherYourNameTag" (in this documentation I used "Weather" as my tag)*



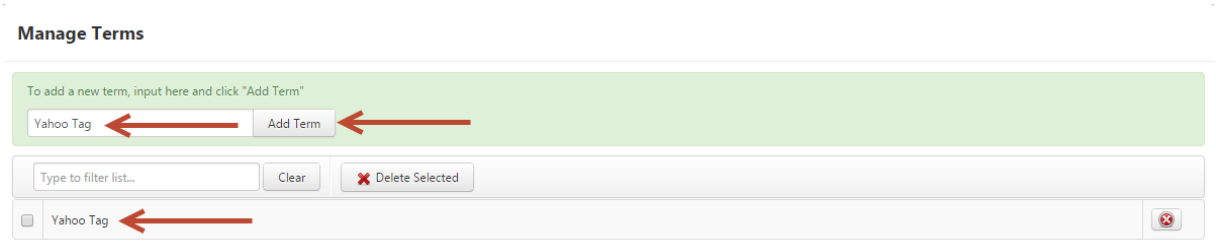
- *Press the **Edit** button in order to get back to editing mode*



- *Select the **Manage Terms** option*



- *Add a new Term by typing the name of the term and press **Add Term**. An example would be "WeatherYourNameTerm" (in this documentation I used "Yahoo Tag" as my term for the "Weather" tag). You can see that for the term of a vocabulary (tag) the naming restriction doesn't apply, although for the tag itself, it applies. Then, you are able to see the new term and then you can press **Done***



- *Press **Save** again in order to save the Model Tag with the new Term*

In Step 1, you've created a new Model Tag that you used to identify all our mashups components. If you ever want to export a mashup from an instance of ThingWorx to another instance of Thingworx, you can do this with the export feature, and use the tag that you created in order to export everything that has tag.

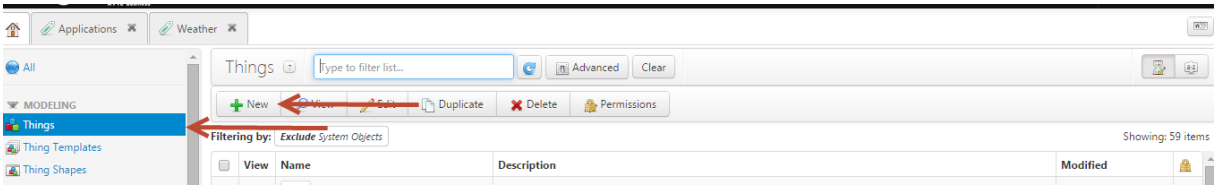
Step 2: Create a simple thing called "WeatherThing"

In this step, you create a thing that you use to store all your data in ThingWorx.

The ThingWorx application development environment operates on an object or thing-driven model, which means that you represent your process in terms of things that have properties and business logic.
In addition to configuring things to represent your physical assets, you can also use things for a variety of other tasks. To facilitate this, all things are based on thing templates.
When you create a thing, you can base it off of your custom template, one of the built-in thing templates, or the generic template. Creating a thing using the generic template creates a thing with no inherited properties. The use of templates makes it easy to add new things and to add or remove properties.

Create a WeatherThing Thing

- *Select the **Things** option from the **Modeling** section in the **Home** tab and click the **New** button*



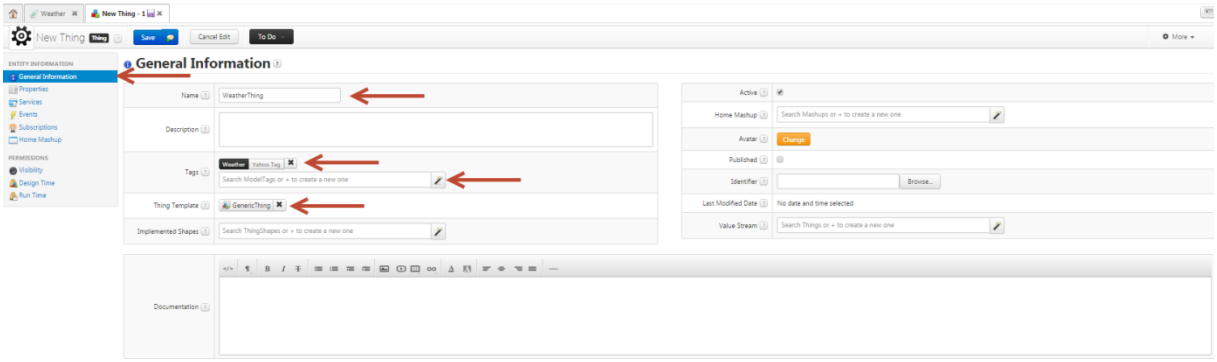
- *Name the new **Thing** "WeatherThing" in the **General Information** section.*

- *Select the tag created in step 1: **Weather Yahoo Tag** from Tags (in every case, this tag can have a different name, depending on what name you've set for it in step 1)*

- *Select **GenericThing** as the Base Thing Template (every Thing in ThingWorx must have assigned to it a Thing Template. If you don't need to create a new Thing Template with specific services, you will use the default, GenericThing template. This default template has all the necessary services that a Thing needs to work. A Thing gets all his characteristics from a Thing Template, so in yout case you will have the minimal characteristics needed for a Thing to work.)*

*The ThingWorx Composer provides an Auto-Complete feature in its search forms. By typing "g" into the **Thing Template** text box it will filter the list to all templates that begin with "g" and make it easier to find the **GenericThing** Template. All the ThingWorx Composer search wizards make it easy to find selections or entities in this manner.*

- *The screen appears as shown below:*



- *After all of these have been created/edited, click the **Save** button to save your new Thing.*

In Step 2, you created a Thing that enables you to log and display the information in and from your mashup. You also use this Thing to create the bindings between the widgets from ThingWorx and the data services from this Thing.

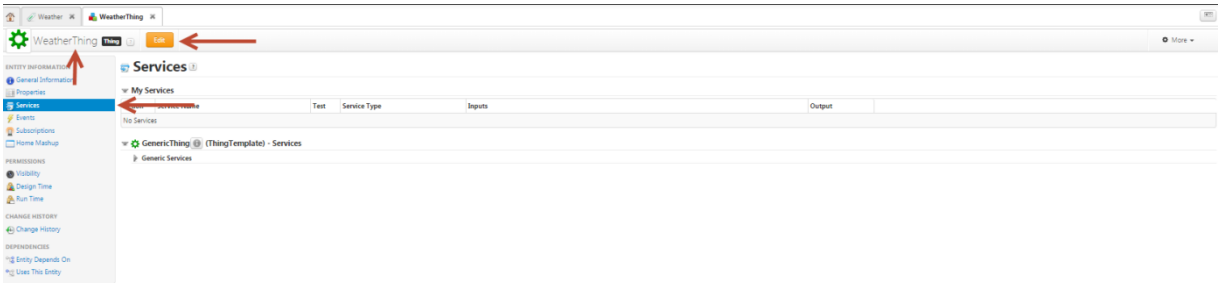
Step 3: Create a service for the thing called "WeatherThing"

In this step, you create a service that will gather hourly weather data from the Yahoo Weather site.

*All **Things** have **Services**. These **Services** are functions that can change the properties of a thing. They are not limited just changing the properties of a Thing but can do anything since they are written using Javascript. A **Service**, like any function has input parameters and can return one output value.*

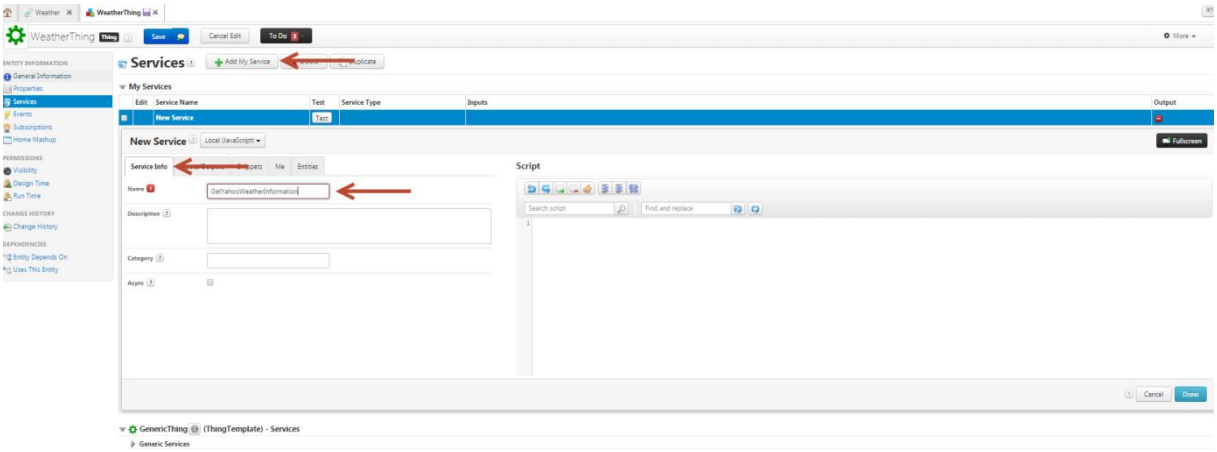
Create a service called GetYahooWeatherInformation

- *Select **Services** under the **Entity Information** section from the **WeatherThing** Thing;*
- *Click the **Edit** button;*



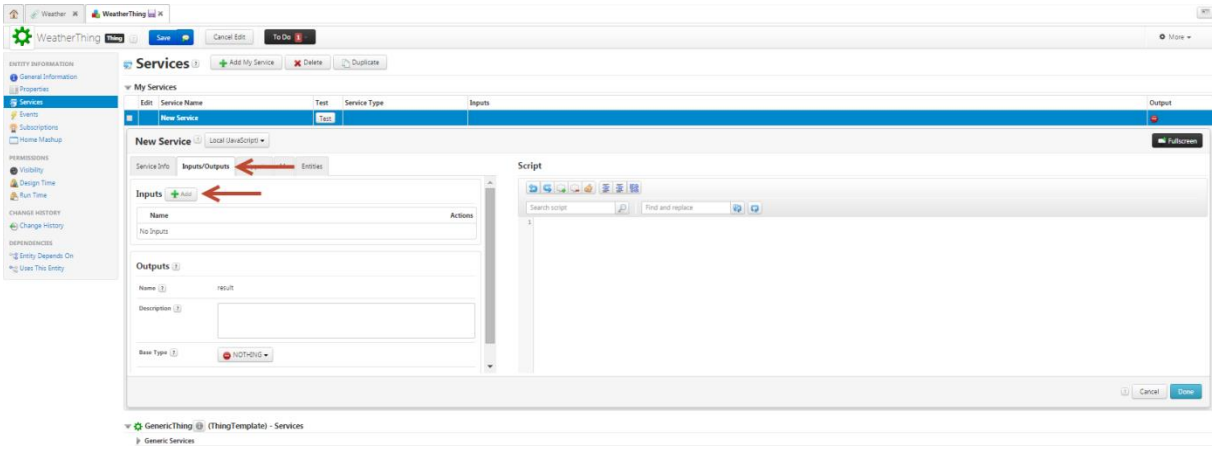
- *Click the **Add My Service** button;*

- *In the **Service Info** tab, name the new service: **GetYahooWeatherInformation;***



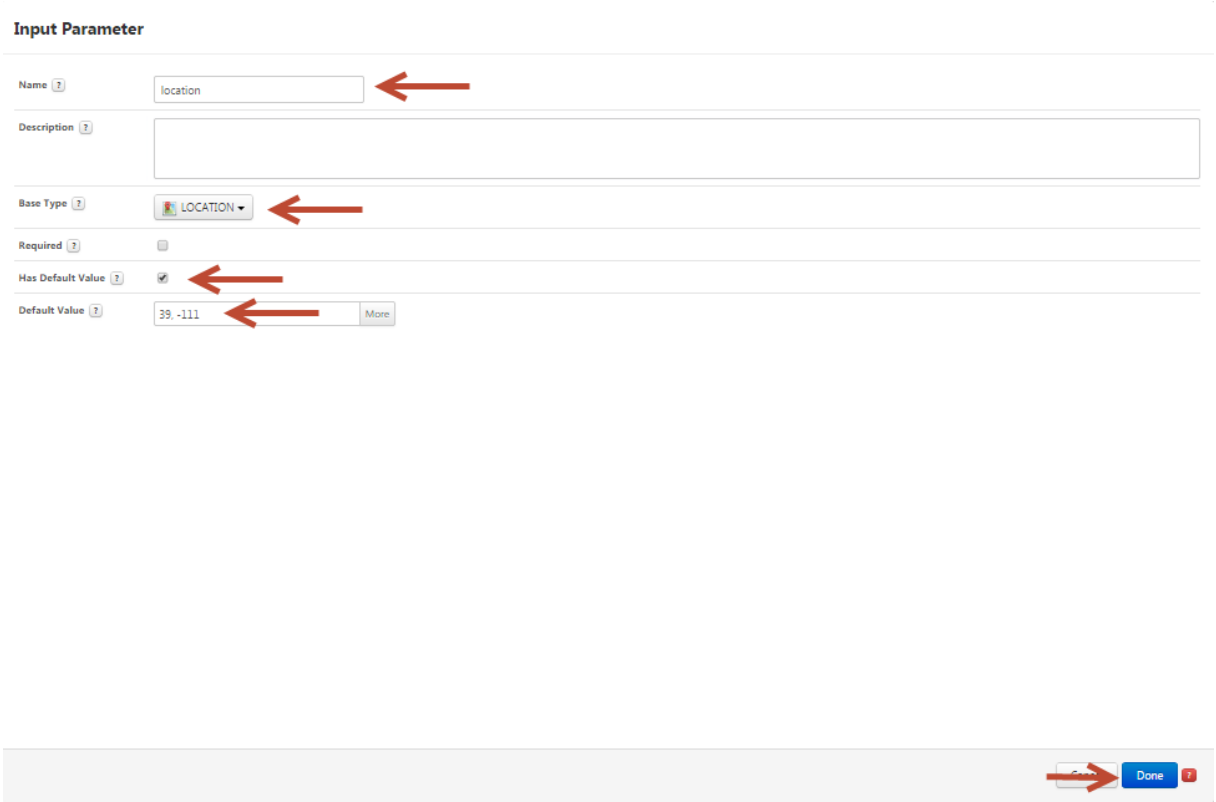
- *In the Inputs/Outputs tab, add one new Input called location. This input enables you to change the location at any time you want, straight from the mashup. With this input you can choose the City for which you want to gather weather data.*

- *Navigate* to the **Inputs/Output** tab.
    - *Add a new input parameter called location by clicking the green **+Add** button:*



    - *The input section will have the following Parameter:*
        - *Name : location , Base Type : location, Has Default Value : Check as true, Default Value : 39, -111 (this is for Ferron, Utah) and click **Done**;*

    - *The information for the input parameters looks like the following :*



- *In the Script section, copy the following text:*

The Open Weather API requires an API Key. You can create yourself one at this website or you can use one that I have created for test 4abc8a3c9f101ccf9c7cfe7cbf3dfed7. You need to append &APPID=4abc8a3c9f101ccf9c7cfe7cbf3dfed7 (or your own api key value) at the end of the Open Weather URL

Free accounts have limited capacity and data availability. To see a list of prices and capabilities for each account visit this website.

```
if(location!=null){
    var test=location+" ";
    var arrayLocations=test.split(",");
    var lat=arrayLocations[0];
    var long=arrayLocations[1];
   var prm = {
        url:
"http://query.yahooapis.com/v1/public/yql?q=SELECT%20*%20FROM%20geo.placefinder%20WHERE%20text%
3D%22"+lat+","+long+"%22%20and%20gflags%3D%22R%22" /* STRING */,
        timeout: 60 /* NUMBER */
    };
    // result: XML
    var xmlPg = Resources["ContentLoaderFunctions"].LoadXML(prm);
    var yahoo = new Namespace('http://www.yahooapis.com/v1/base.rng');
    // CreateInfoTableFromDataShape(infoTableName:STRING("InfoTable"),
dataShapeName:STRING):INFOTABLE(YahooWeatherFeed)
woeid =xmlPg.results.Result.woeid;

}
var params = {
    url: "http://xml.weather.yahoo.com/forecastrss?w=" +woeid + "&u=f" /* STRING */,
    timeout: 60 /* NUMBER */
};

// result: XML
var xmlPage = Resources["ContentLoaderFunctions"].LoadXML(params);
var yweather = new Namespace('http://xml.weather.yahoo.com/ns/rss/1.0');
```

```
var params = {
    infoTableName : "InfoTable",
    dataShapeName : "YahooWeatherFeed"
};

// CreateInfoTableFromDataShape(infoTableName:STRING("InfoTable"),
dataShapeName:STRING):INFOTABLE(YahooWeatherFeed)
var result = Resources["InfoTableFunctions"].CreateInfoTableFromDataShape(params);

for each (var item in xmlPage.channel) {
    var row = new Object()
    row.Link = xmlPage.channel.item.link;
    row.Title = xmlPage.channel.item.title;
    row.WeatherCondition = "http://l.yimg.com/a/i/us/we/52/" +
xmlPage.channel.item.yweather::condition.@code + ".gif";
    row.WeatherTemp = parseInt(xmlPage.channel.item.yweather::condition.@temp.toString());
 row.WeatherHumidity = parseInt(xmlPage.channel.yweather::atmosphere.@humidity.toString());
    row.WindSpeed = xmlPage.channel.yweather::wind.@speed.toString() + " " +
xmlPage.channel.yweather::units.@speed.toString();
    row.WindDirection = xmlPage.channel.yweather::wind.@direction.toString();
    row.Sunrise = xmlPage.channel.yweather::astronomy.@sunrise.toString();
    row.Sunset = xmlPage.channel.yweather::astronomy.@sunset.toString();
    row.City = xmlPage.channel.yweather::location.@city.toString();
}
var windDValue=parseInt(xmlPage.channel.yweather::wind.@direction.toString());

if(windDValue>0&&windDValue<45){
    row.WindDirection="NNE";
}
else if(windDValue==0){
    row.WindDirection="N";
}
else if(windDValue==45){
    row.WindDirection="NE";
}
else if(windDValue>45&&windDValue<90){
    row.WindDirection="ENE";
}
else if(windDValue==90){
    row.WindDirection="E";
}
else if(windDValue>90&&windDValue<135){
    row.WindDirection="ESE";
}
else if(windDValue==135){
    row.WindDirection="SE";
}
else if(windDValue>135&&windDValue<180){
    row.WindDirection="SSE";
}
else if(windDValue==180){
    row.WindDirection="S";
}
else if(windDValue>180&&windDValue<225){
    row.WindDirection="SSW";
}
else if(windDValue==225){
    row.WindDirection="SW";
}
else if(windDValue>225&&windDValue<270){
    row.WindDirection="WSW";
}
else if(windDValue==270){
    row.WindDirection="W";
}
else if(windDValue>270&&windDValue<315){
    row.WindDirection="WNW";
}
else if(windDValue==315){
    row.WindDirection="NW";
}
else if(windDValue>315&&windDValue<360){
    row.WindDirection="NNW";
}
else if(windDValue==360){
    row.WindDirection="N";
}
```

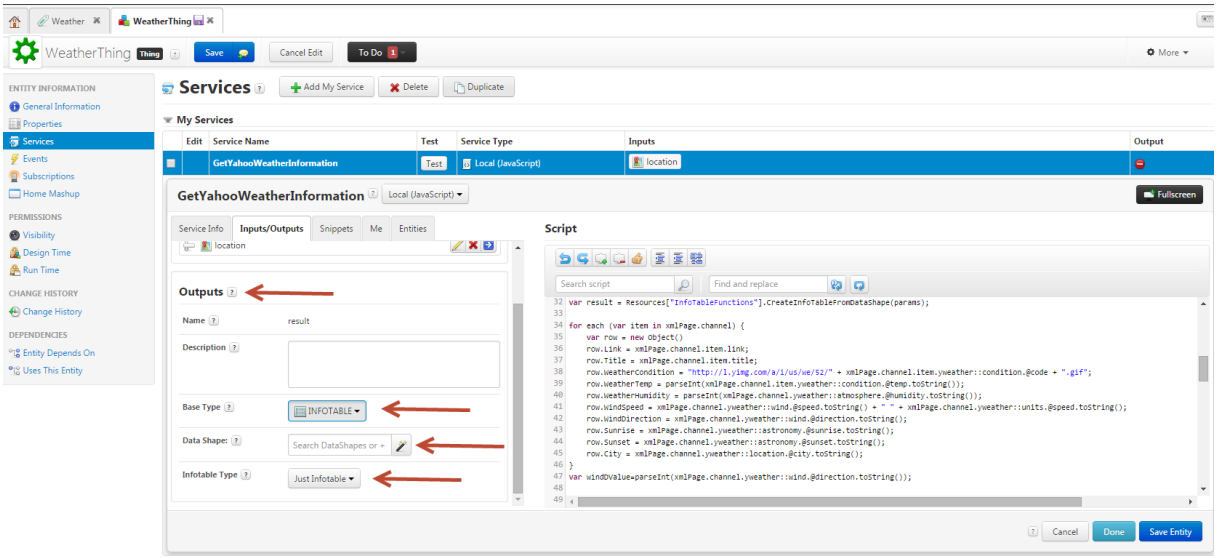
```
var cityName=xmlPage.channel.yweather::location.@city;
//Modify the open weather url here if you are using your own API key
var params = {
        url:
        "http://api.openweathermap.org/data/2.5/weather?q="+cityName+"&type=accurate&mode=xml&A
        PPID=4abc8a3c9f101ccf9c7cfe7cbf3dfed7"
        /* STRING */,
        timeout: 60 /* NUMBER */
};
var xmlPage = Resources["ContentLoaderFunctions"].LoadXML(params);
var isPrecipitation=xmlPage.precipitation.@mode;
if(isPrecipitation=="no"){
    row.Precipitation="No Precipitations";
}
else
{
    row.Precipitation=xmlPage.precipitation.@type+" "+xmlPage.precipitation.@value+"
"+xmlPage.precipitation.@unit;
}

row.WindDirection=xmlPage.wind.direction.@name;
result.AddRow(row);
```

This is a piece of JavaScript code that retrieves weather data information from the Yahoo Weather site like temperature, humidity, precipitation, wind speed, wind direction, sunrise and sunset.

- *After you've added this input and the text in Script, you need to edit the Outputs section. On the Outputs section, select **Base Type**: INFOTABLE, **Infotable Type**: Just InfoTable and now, you also need to add a **Data Shape**. You don't yet have a Data Shape, so you need to create one.*

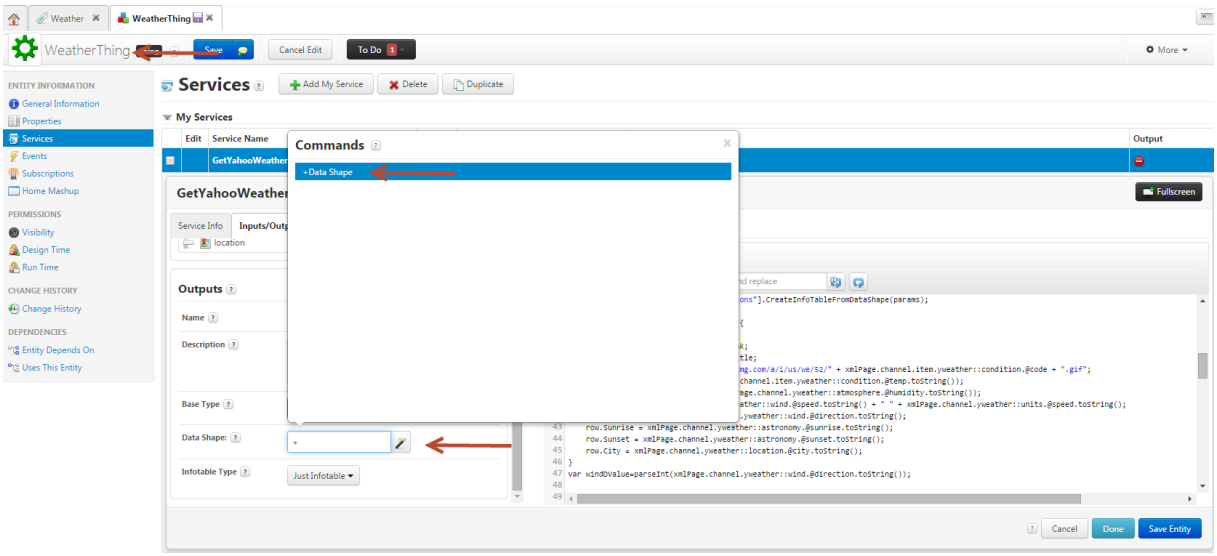
DataShapes are how you describe the data parts of your model. A DataShape is a named set of field definitions and related meta-data. Each field in a DataShape has a data type. DataShapes help you to create applications, because when you consume data, the application has built-in knowledge of how to represent the data set because of the shape definition. DataShapes are also used wherever you need to describe a data set. For example, when you define an InfoTable output for a Service implementation, you use a DataShape to describe the output result set.

- *The screen appears as shown below:*



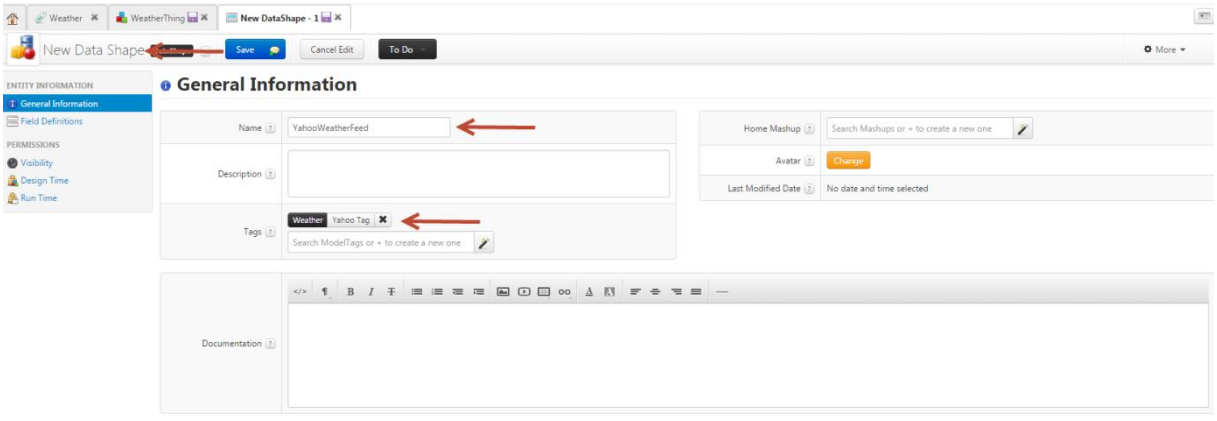
*What's an **InfoTable**? An **InfoTable** is an in memory ordered collection of objects, each of which is described by the same DataShape. It is similar to an array of objects but Thingworx provides many utility functions to manipulate info tables so that they can provide a common model for any form of external data.*

- *To add a new Data Shape, just press the "+" sign inside the Data Shape field. It will open a screen where you can select to add a new Data Shape. The name restriction will also apply to the Data Shape, so remember to use one word when you will name the Data Shape. The screen appears as shown below:*



- *Name the new **Data Shape** "YahooWeatherFeed" in the **General Information** section*
- *Select the tag created in step 1: **Weather Yahoo Tag** from Tags*

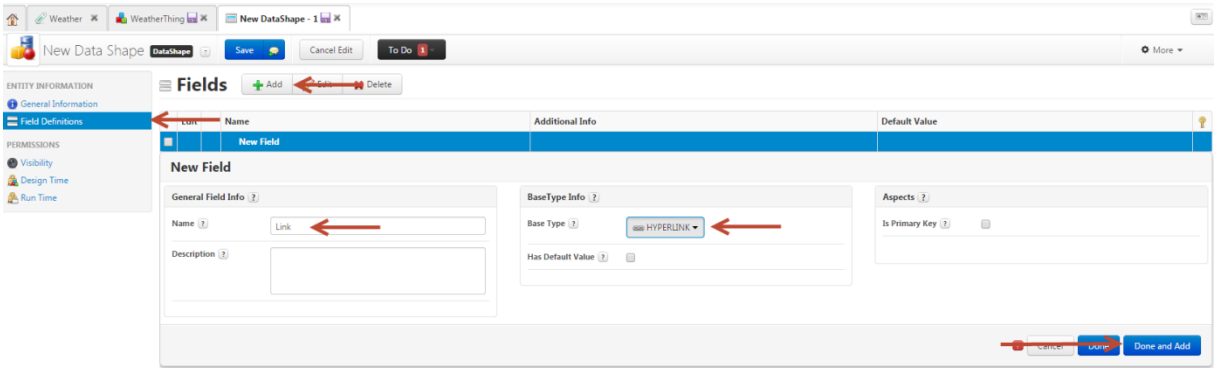
- *The screen appears as shown below:*



- *Select **Field Definitions** under the Entity Information section from the **Data Shape***

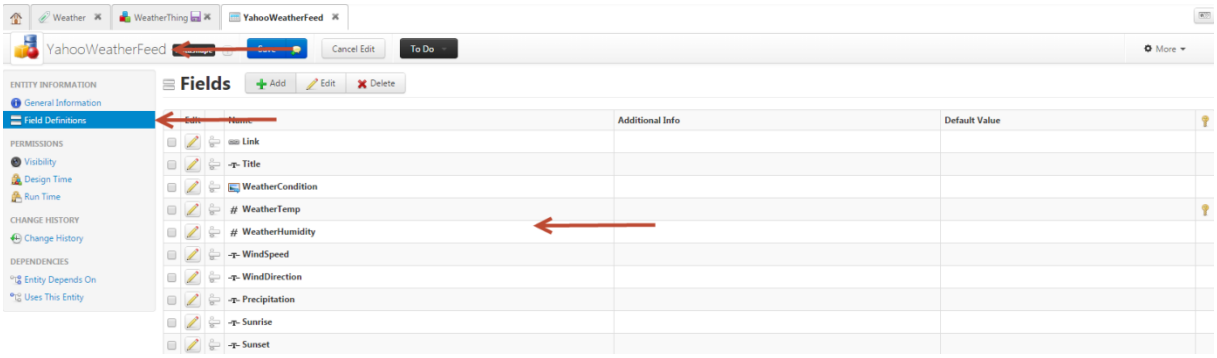
- *Click the **Add** button*

- *Complete the parameters as follows: Name = "Link", Base Type = HYPERLINK*
- *Click the **Done and Add** button. The screen appears as shown below:*

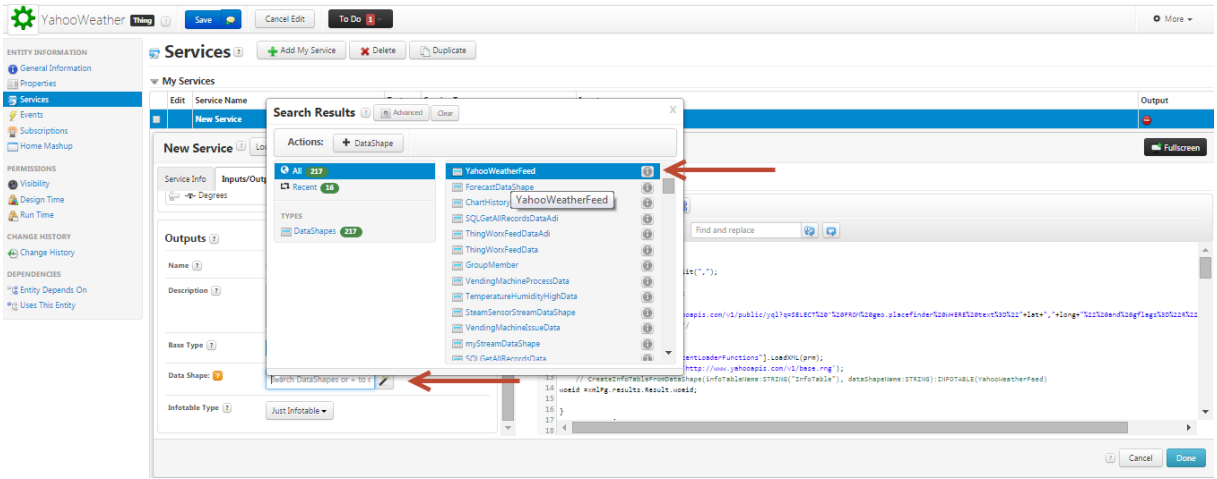


- *Complete the parameters as follows: Name = "Title", Base Type = STRING;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "WeatherCondition", Base Type = IMAGELINK;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "WeatherTemp", Base Type = Number;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "WeatherHumidity", Base Type = Number;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "WindSpeed", Base Type = STRING;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "WindDirection", Base Type = STRING;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "Sunrise", Base Type = STRING;*
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "Sunset", Base Type = STRING;*

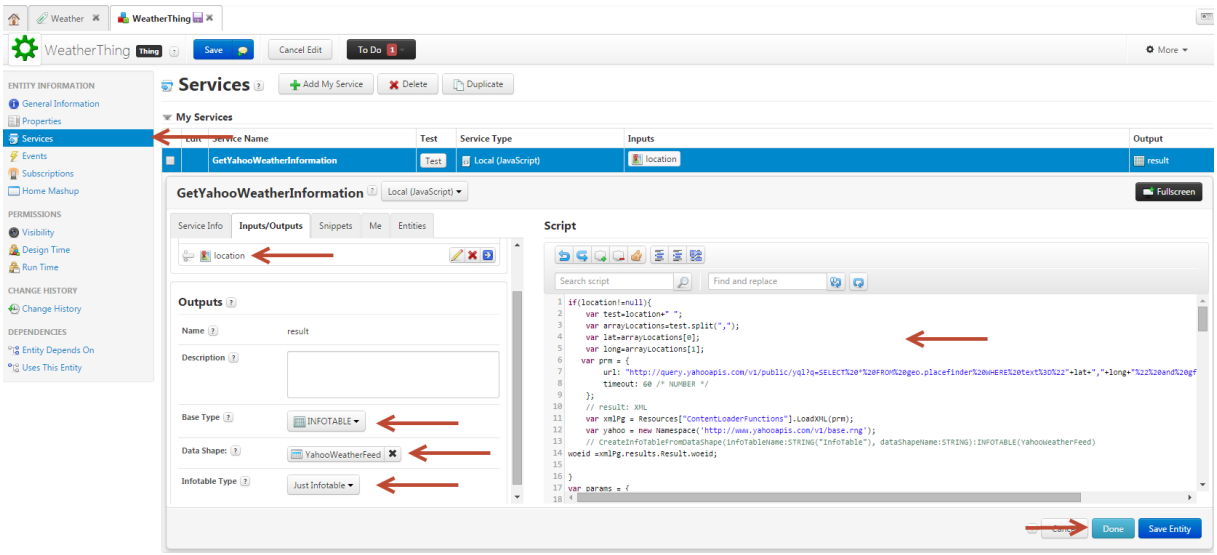
- *Click the **Done and Add** button.*
- *Complete the parameters as follows: Name = "City", Base Type = STRING;*
- *Click the **Done** button. After that click **Save** button to save all your fields in the new Data Shape. The screen appears as shown below:*



- *Go to the Outputs section and set the newly created Data Shape (YahooWeatherFeed) as the DataShape. The screen appears as shown below:*



- *Click **Done** to finish editing your new Service. The screen appears as shown below:*



- *Don't forget to save your Service and Thing by pressing the **Save** button;*

In Step 3, you created a service called GetYahooWeatherinformation for our Thing. You will use this service to get hourly weather data information from the Yahoo Weather site

Step 4: Create a mashup called "WeatherMashup"

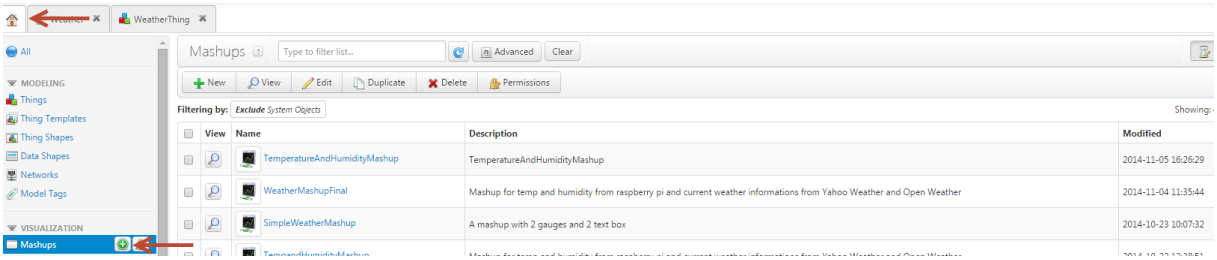
In this step, you create a mashup to display the weather data from the Yahoo Weather site. From the mashup you can choose for what city you want to see the current weather data.

Create a WeatherMashup mashup

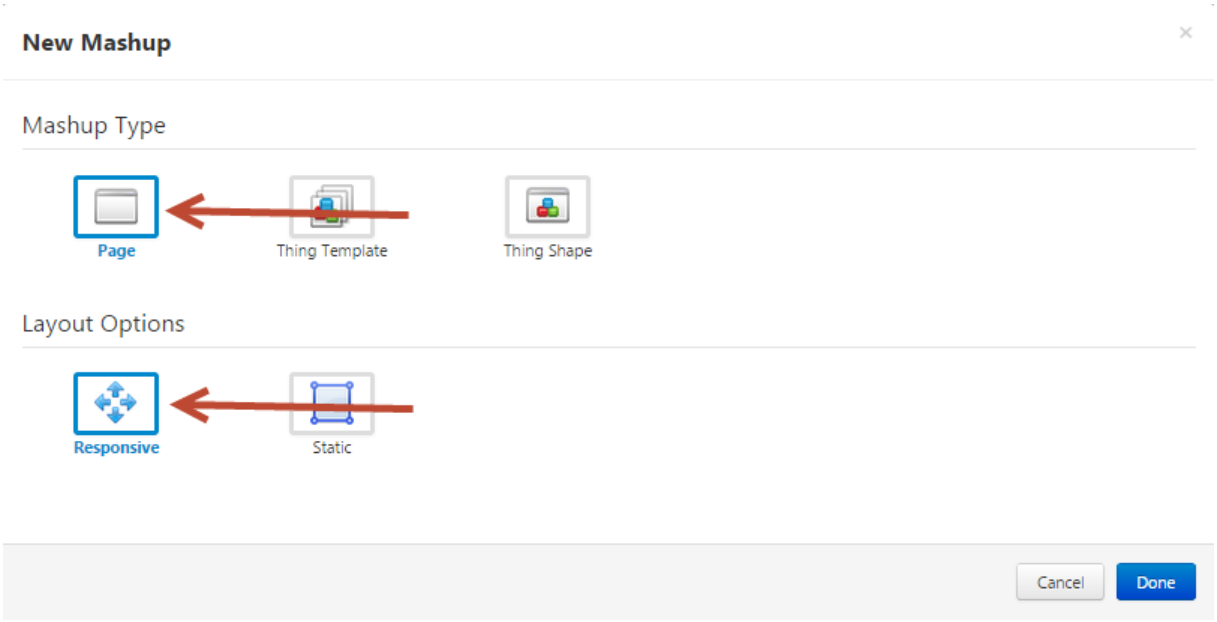
Now that your YahooWeatherThing has been created, you need to build a simple mashup to display the informations stored in his service.

Mashup is the term used for a web page that allows you to visualize and interact with the model you have defined. Mashups usually integrate information from multiple sources bringing information together, potentially exposing relationships or giving new meaning to your model data.

- *First navigate to the **Home** tab;*
- *On the left panel, go over the **Mashups** entry in the **Visualization** section, and then click the green **+** button as shown below:*

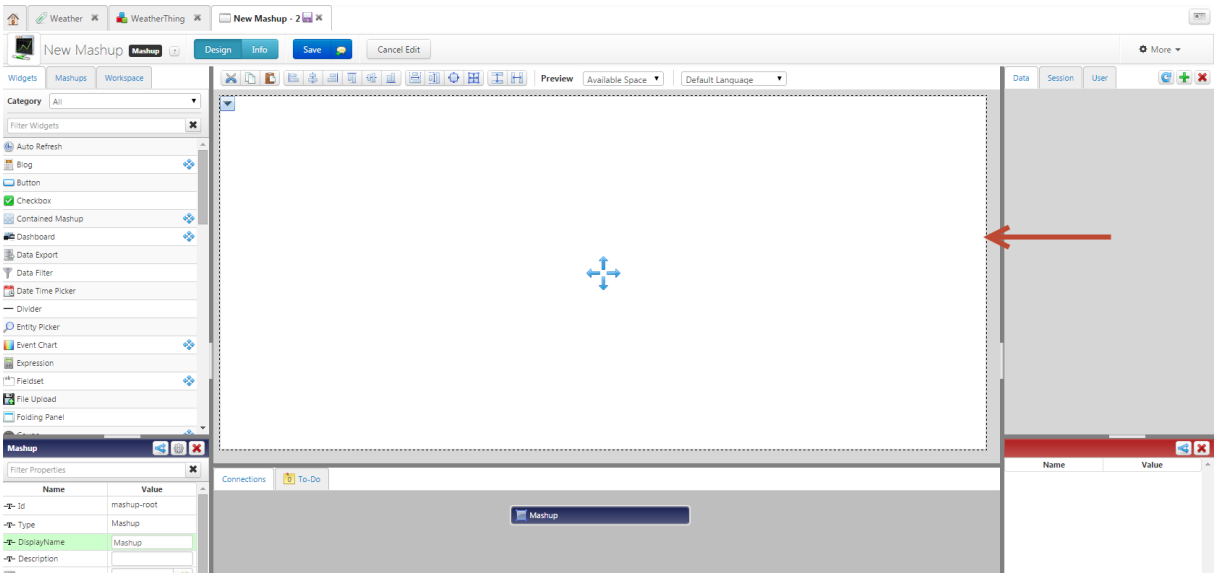


- *Leave the default **Mashup Type** of Page*
- *Select Responsive as the **Layout Option***



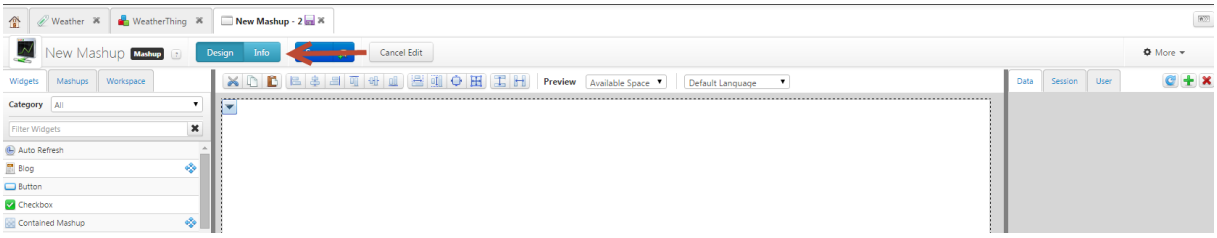
- *A mashup can be "**Responsive**" or "**Static**". A responsive mashup means that it will always fill to the resolution of the display, never leaving any unused space around the mashup. A static mashup is statically sized to the dimensions that you define. When displayed in a lower resolution, it will have scroll bars, in a higher resolution it will leave unused space around the mashup;*

- *After clicking **Done** you will have a blank canvas on which to build your Mashup, as show below:*



- *The mashup will automatically open in the **Design** state, here you have the following information:*
    - *On the left side you have the **Widgets** side of the mashup. You have 3 tab's(Widgets, Mashups, Workspace) in which you can see the available widgets and mashup's;*
    - *On the middle you have the work area (workspace) where you drag and drop your widgets + services to bind the widgets;*
    - *On the bottom you have the 2 tabs(**Connections** and **To-Do**) that helps you see if you did the correct bindings and if you forgot to bind something from the workspace;*
    - *On the right side you have the Services part of the mashup;*

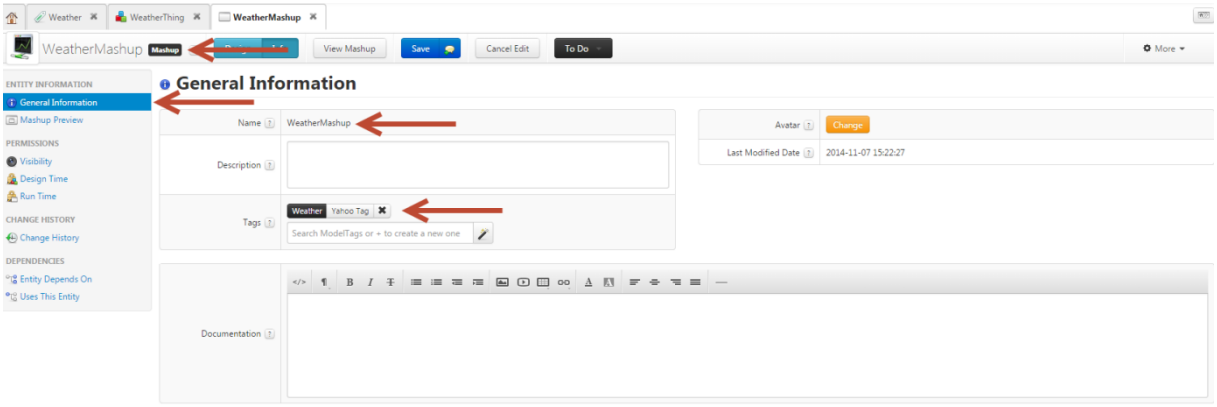
- *After this, press the **Info** button, next to **Design;***



- *Name the new Mashup "WeatherMashup" in the **General Information** section;*

- *Select the tag created in Step 1: **Weather Yahoo Tag** from Tags and save it*

- *The screen appears as shown below:*



Add widgets to the WeatherMashup

Now that your WeatherMashup has been created, you add widgets to it in order to display the hourly weather data information from the Yahoo Weather site. You need to add two layout widgets, one panel, a button widget, a Google Location Picker widget and a Google Map widget. Their purpose will be explained below.

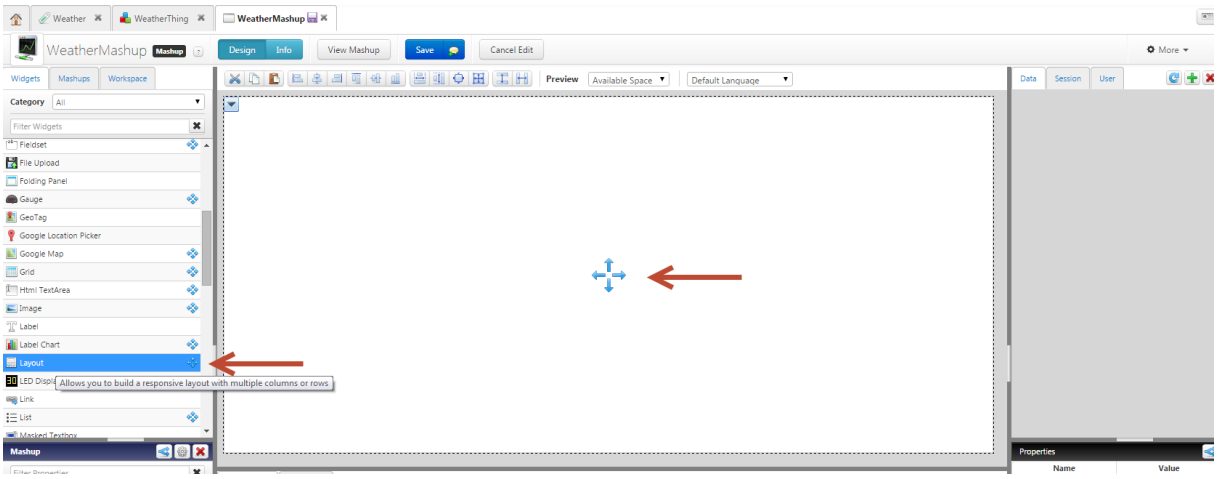
*A **Widget** is a visual element you can place in a **Mashup** that understands how to display information formatted in a useful way. Table **Widgets** know how to format the rows of data in your model as an HTML table. Map **Widgets** know how to display your position on a map from your latitude and longitude. **Mashups** are composed of multiple widgets that work together to display information.*

- *First, you the two layout widgets.*

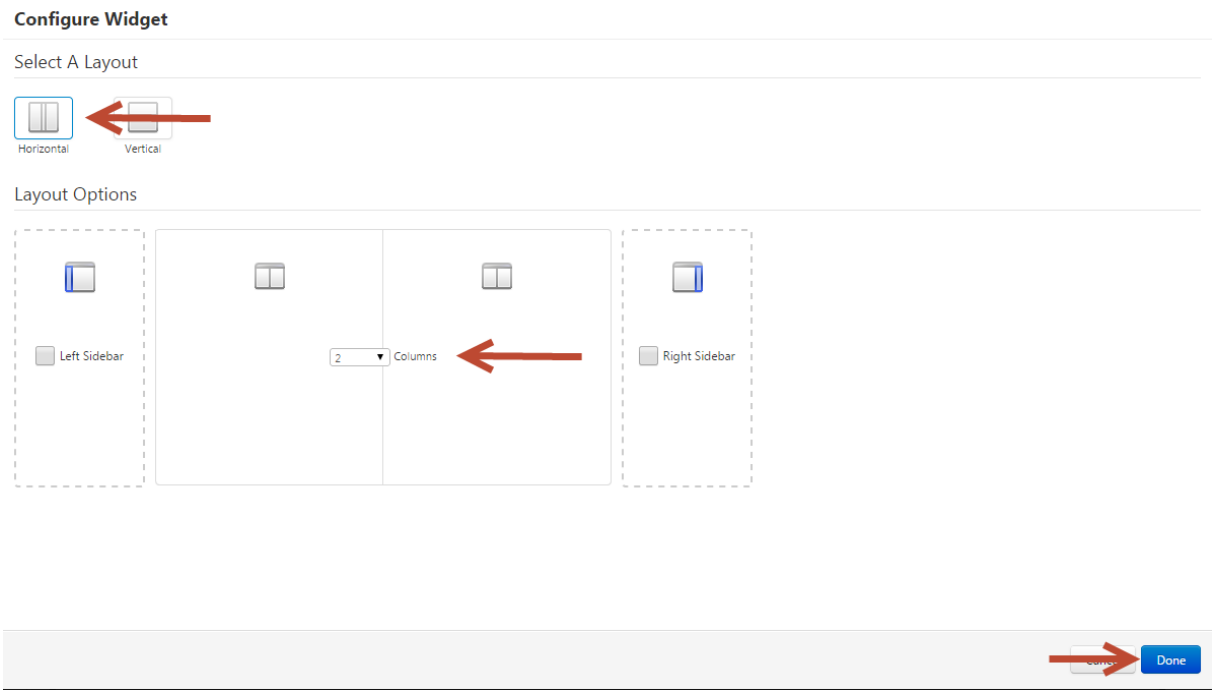
*The **Layout** Widget is a responsive container that allows you to separate a responsive container into sections. Layouts can be placed inside the responsive containers of other layouts to create more even sections.*
The Header and Footer have constant heights with responsive widths, while the sidebars have a constant width and responsive heights. The Columns and rows have both responsive widths and heights, but you can change the percent of available space they will occupy relative to the space not being occupied by headers, footers, and sidebars.

So, use the layout widget in order to "split" your mashup into more responsive areas.

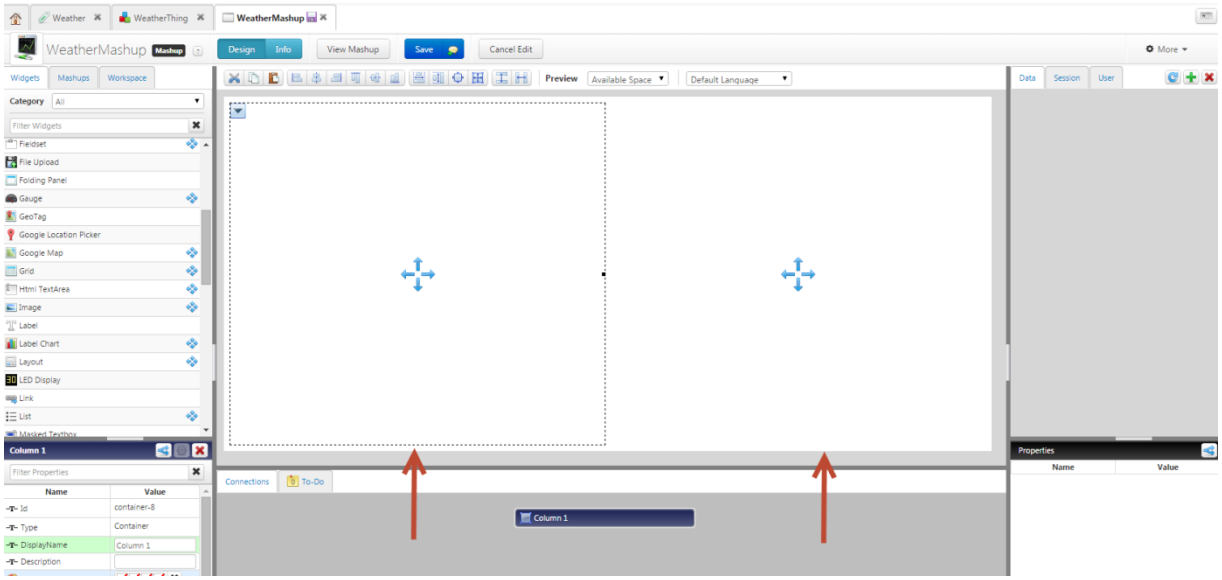
- *From the **Widgets** panel on the top left scroll down to the **Layout** widget and drag and drop it onto your Mashup as shown below:*



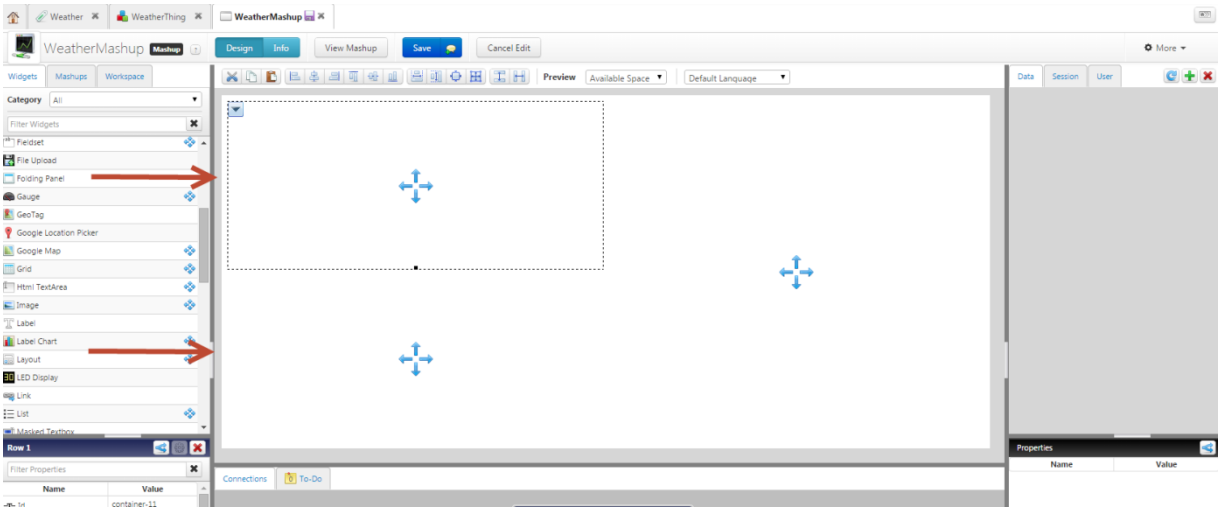
- *After you've added your layout widget, a screen opens in which you configure your widget. From the "Select A layout" select* ***Horizontal****, and from the "Layout Options" select 2 columns and press* ***Done,*** *like in the screen below:*



- *You will see your mashup with a layout widget divided in 2 rows like in the screen below:*



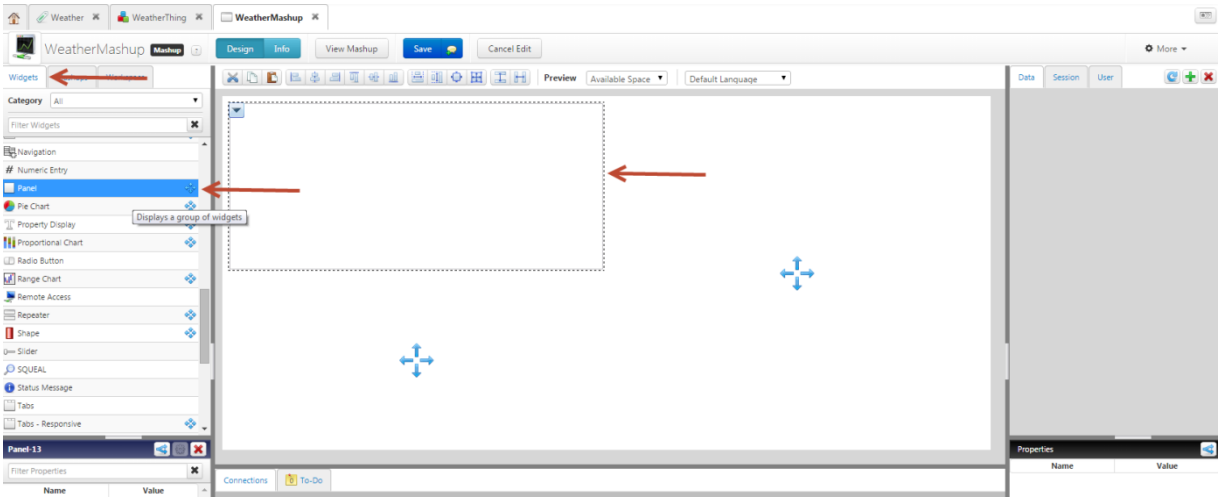
- *Next, add another layout widget in the left column. A layout widget that is Vertical and has 2 rows. After you've added this widget, the mashup will look like in the screen below:*



- *Don't forget to save it by pressing the **Save** button*
- *Next, in the first row, add 3 widgets (a panel, a google location picker and a button) as follow:*

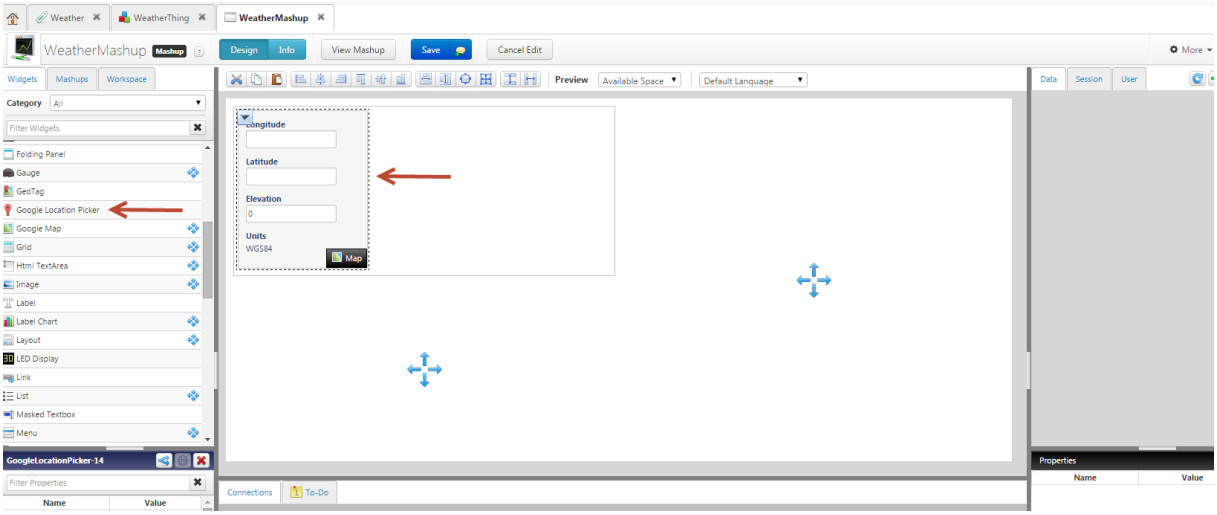
A. *Adding a **Panel** widget – you use this to store your other 2 widgets (the button widget and the google location picker widget). The panel is responsive, so this means that it takes all the space from the column. From the **Widgets** panel on the top left, scroll down to the **Panel** widget and drag and drop it on your first row inside the mashup as shown below:*

*The **Panel** Widget is a container. You can place other Widgets into a Panel and move them around as a group. You can also use a Style to change the background color of a Panel to add contrast to your Mashup. All Widgets placed in the Panel are relative to the Panel location.*



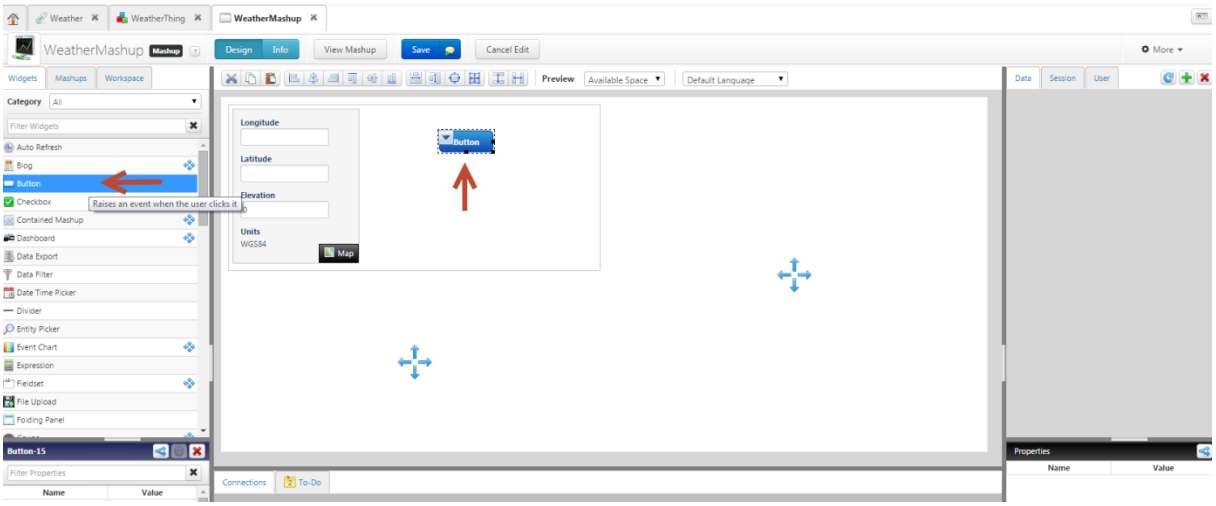
B. *Adding a **Google Location Picker** widget – you use this widget search for the City for which you want to have weather data. From the **Widgets** panel on the top left, scroll down to the **Google Location Picker** widget and drag and drop it onto your panel as shown below:*

The Google Location Picker Widget is a control that allows a user to to either enter a location via WGS84 format of latitude, longitude, and elevation, or use a Google map to pinpoint a location. That location can then be bound to other Widgets or Services as needed for your solution.

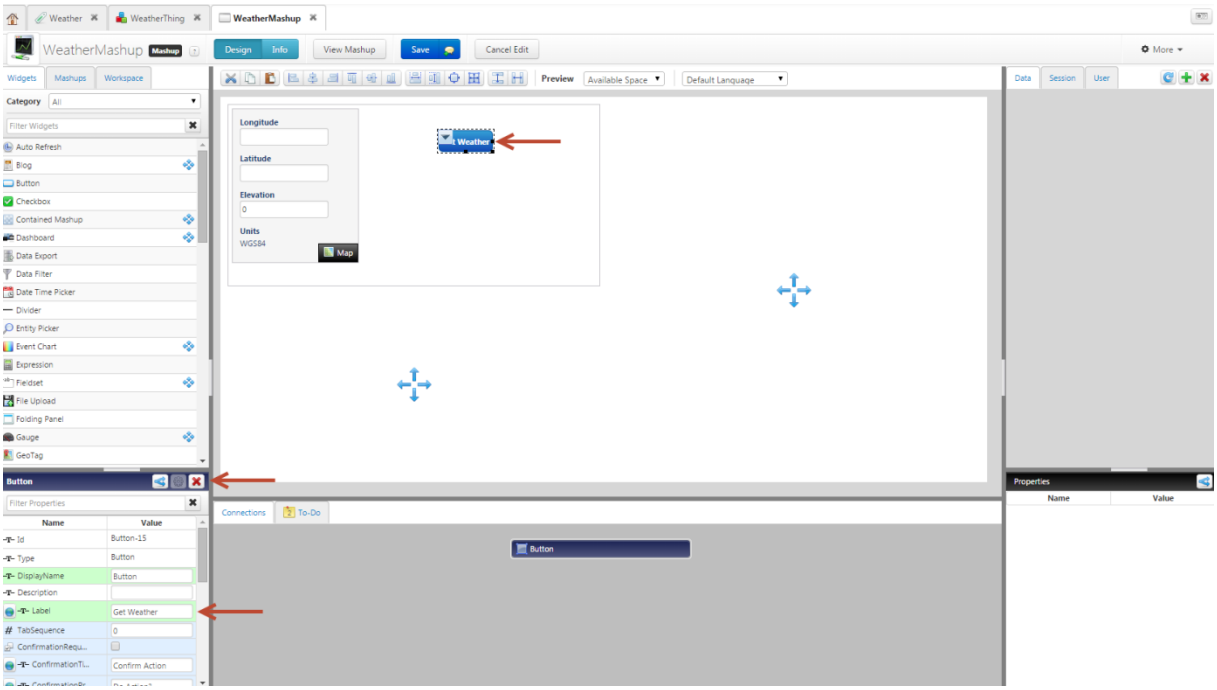


C. *Adding a **Button** widget – you use this to get the weather data. From the **Widgets** panel on the top left, scroll down to the **Button** widget and drag and drop it onto your panel as shown below:*

*The Button widget triggers an **Event** when clicked.*

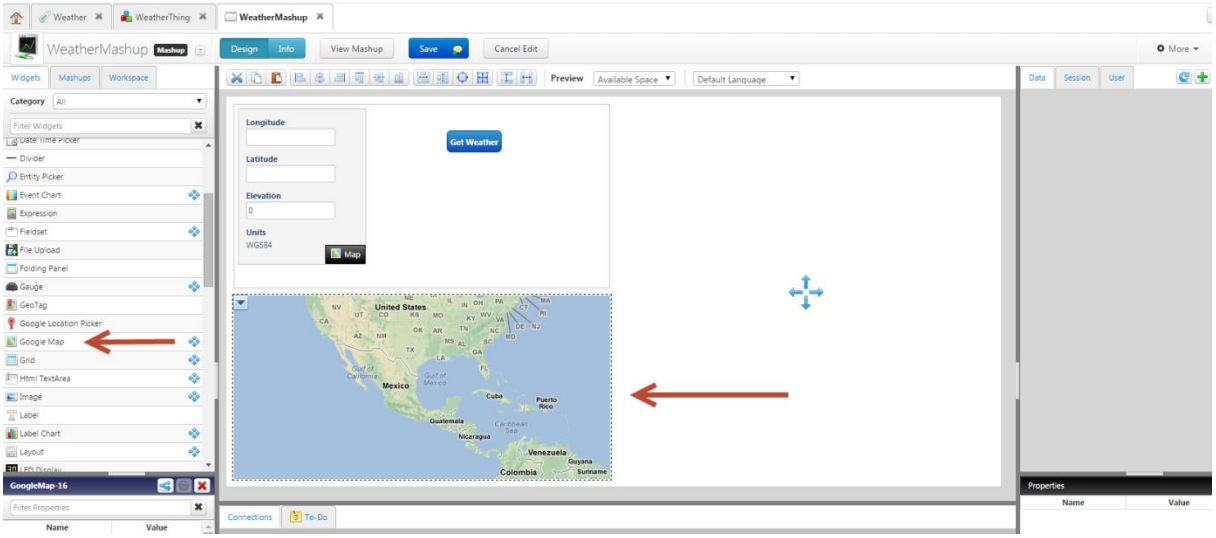


*Next, you modify the properties of the button widget. You can see a widgets property as soon as you click on it. The properties are shown in the bottom left side of the mashup. You modify the button's Label property as follows: Label = Get Weather. By this, you are modifying the name of the Button in **Get Weather** as shown below:*



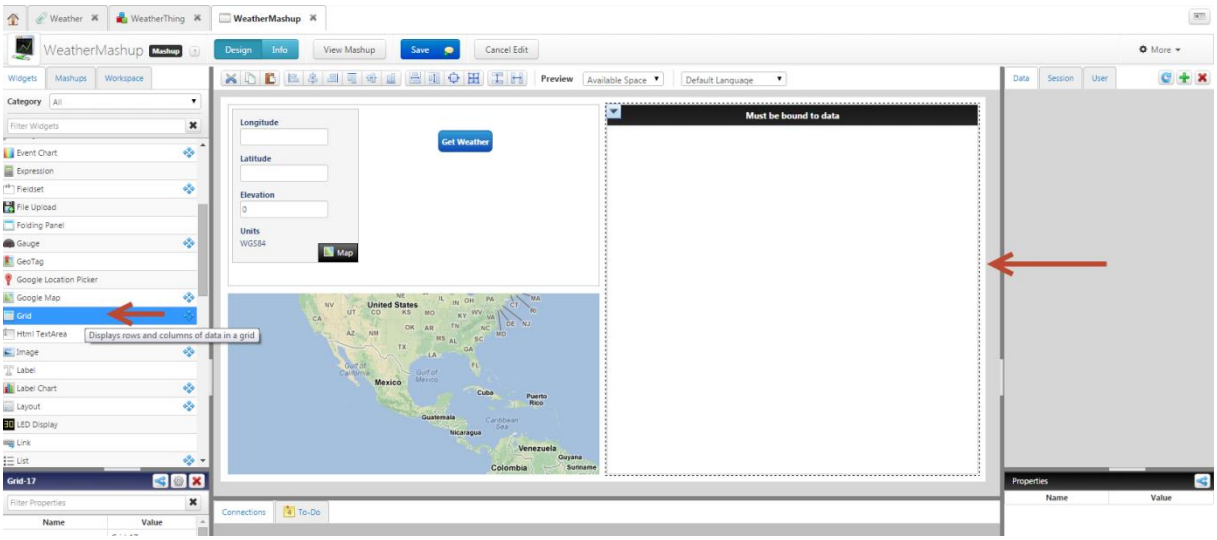
- *Don’t forget to save it by pressing the **Save** button;*
- *Next, in the second row, you add a **Google Map** widget. You use this widget in order to see on the map where the city is. The **Google Map** widget is responsive, so this means that it takes all the space from the row. From the **Widgets** panel on the top left, scroll down to the **Google Map** widget and drag and drop it into the row as shown below:*

The Google Map Widget is a control that allows a user to either enter select locations or see locations of other Things. It is capable of tracking a Thing location history or just marking the current location. It can be used to select sites based on location (such as pick a plant) and update other components on the web page based on the selection.



- *Next, on the right side of the mashup, add a **Grid** widget. You use this widget to see the current weather data information from the Yahoo Weather site. The **Grid** widget is responsive, so this means that it takes all the space from the row. From the **Widgets** panel on the top left, scroll down to the **Grid** widget and drag and drop it into the row as shown below:*

The Grid Widget allows you to display any data set in a grid view. The Grid supports many different ways to render the data, including color contexting by column and rendering images in a cell. You can configure the basic renderings, including column order, which columns to view, column headings, and state formatting through the Grid Widget's configuration dialog, using the Configure Grid Columns selection. The Grid supports column resizing in the runtime environment as well as data sorting by clicking on a column header.



- *Don't forget to save it by pressing the **Save** button;*

Step 5: Add GetYahooWeatherInformation service to the WeatherMashup and bind the data

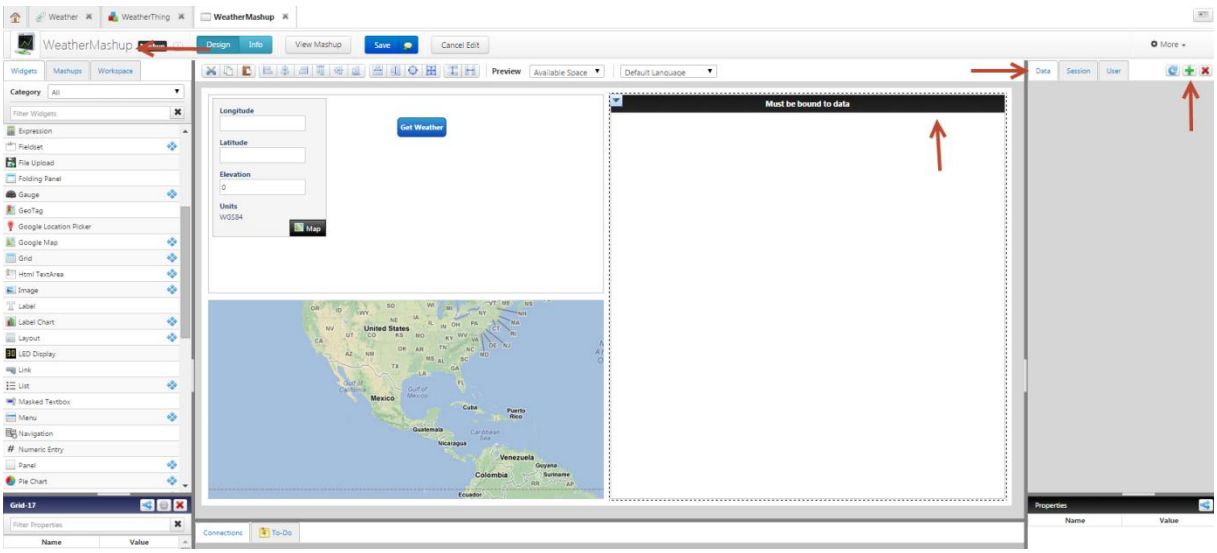
In this step:

- *You apply the "GetYahooWeatherInformation" service created inside the WeatherThing to the newly created mashup called "WeatherMashup";*
- *You bind it to the widgets from the mashup so you can see the weather data information displayed on the widgets;*
- *You learn how to add a new data service inside a mashup and how to bind data services to widgets;*
- *You find out how from the mashup, you can choose for any city to see the current weather data in that city.*

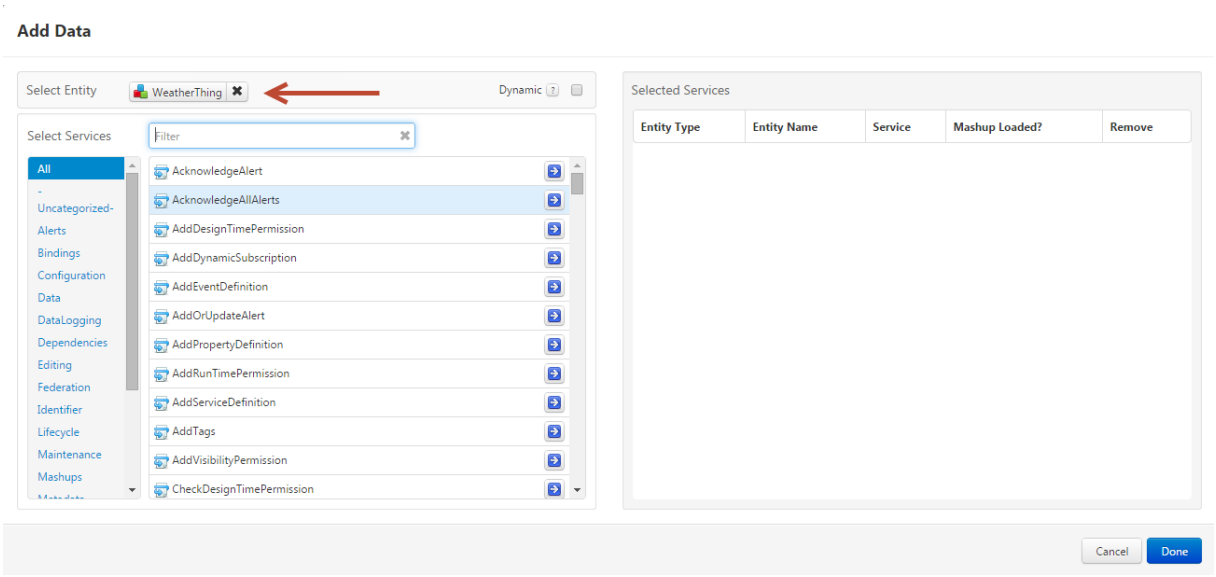
Add GetYahooWeatherInformation service to the WeatherMashup

ThingWorx Data Services are what you call to get data into a Mashup and then render the data using a visualization widget. To render any data in a Mashup, you need to add one or more Data Services to the Mashup. One Data Service can feed multiple visualization widgets. This minimizes the amount of data calls to the server. Most widgets are configurable so that only the columns of data from the Data Service that you wish to see in that widget are shown.

- *In our mashup, Click the green **+** button next to the **Data** tab on the top right hand side of the mashup Editor as shown below:*



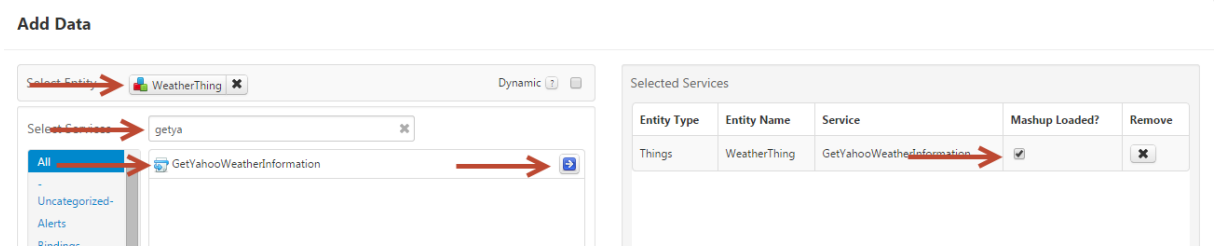
- *In the **Search Entities** textbox of the pop-up window that appears type "Weather" and the WeatherThing thing you have just created appears at the top of the filtered list. Select it as shown below:*



- *In the **Select Services** textbox type "getya" to filter the exposed services for the WeatherThing. Click on the blue arrow button on the right hand side of the GetYahooWeatherInformation service to add it to the Selected Services list.*

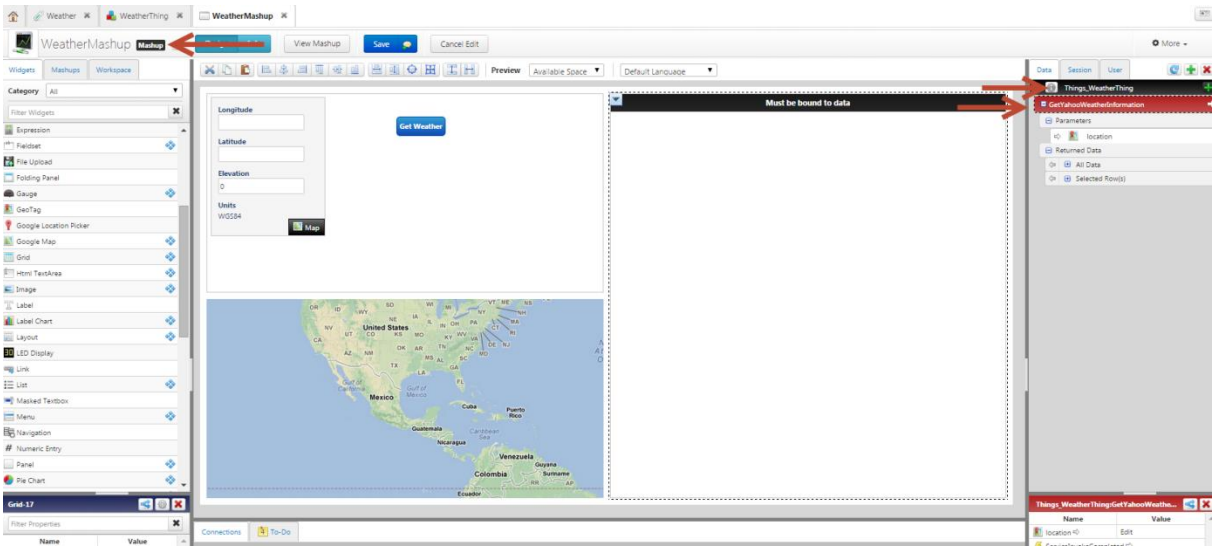
*Why do I have to choose a **Service** to access the data of my **Thing**? At runtime, your **Thing's** data must be transported to your web browser. Your web browser makes a service call to request the current service values of your **Thing** to display in your **Widgets** and if you modify that data, you will have to make another service call to save it well. Any service you define on a Thing can be bound to and called from a **Widget**.*

- *Check the "**Mashup Loaded?"** checkbox. This will cause the GetYahooWeatherInformation Service to be called when your mashup page is first opened.*



*What does "Mashup Loaded?" really do? It allows you to make a **Service** call as soon as all of the Widgets on your page have finished loading. Usually, this service calls job is to fetch the data to initially populate your Widgets.*

- *Click on the **Done** button and you now see this service listed in the **Data** tab of the Mashup Editor*

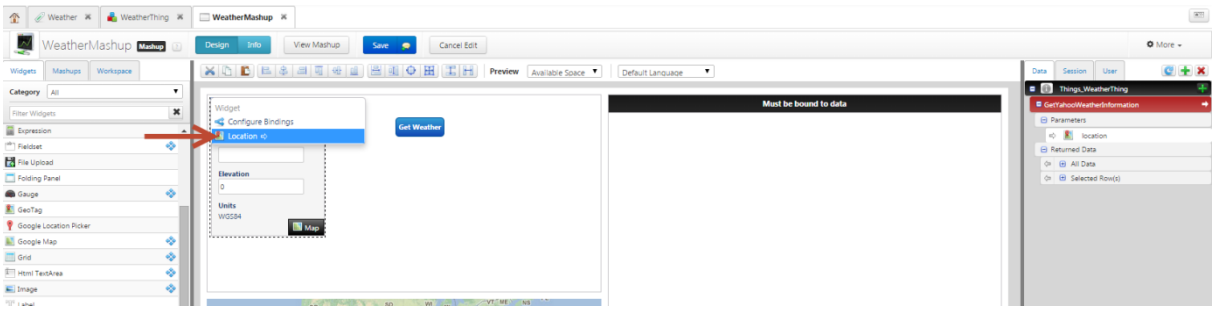


Bind the data between the widgets and the service

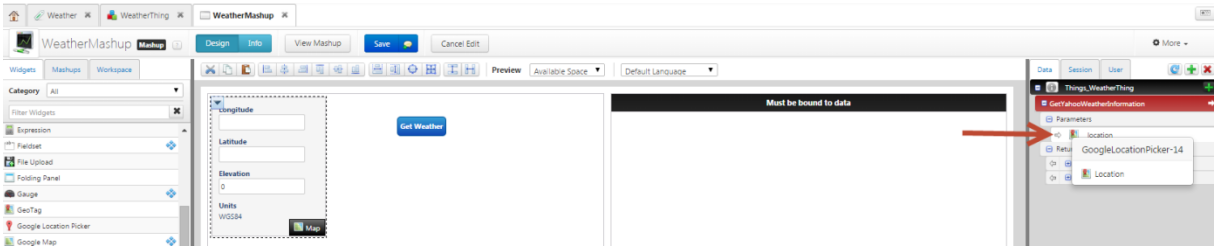
You make the binding in 4 steps:

1 - Bind the Google Location Picker widget to the GetYahooWeatherInformation to choose a city to retrieve weather data from;
2 - Bind the Button widget to the GetYahooWeatherInformation service to retrieve the weather data;
3 - Bind the Grid widget to the GetYahooWeatherInformation service to view the weather data inside the grid widget;
4 - Bind the Google Location Picker widget to the Google Map widget to view the location of the City inside the Google Map widget;

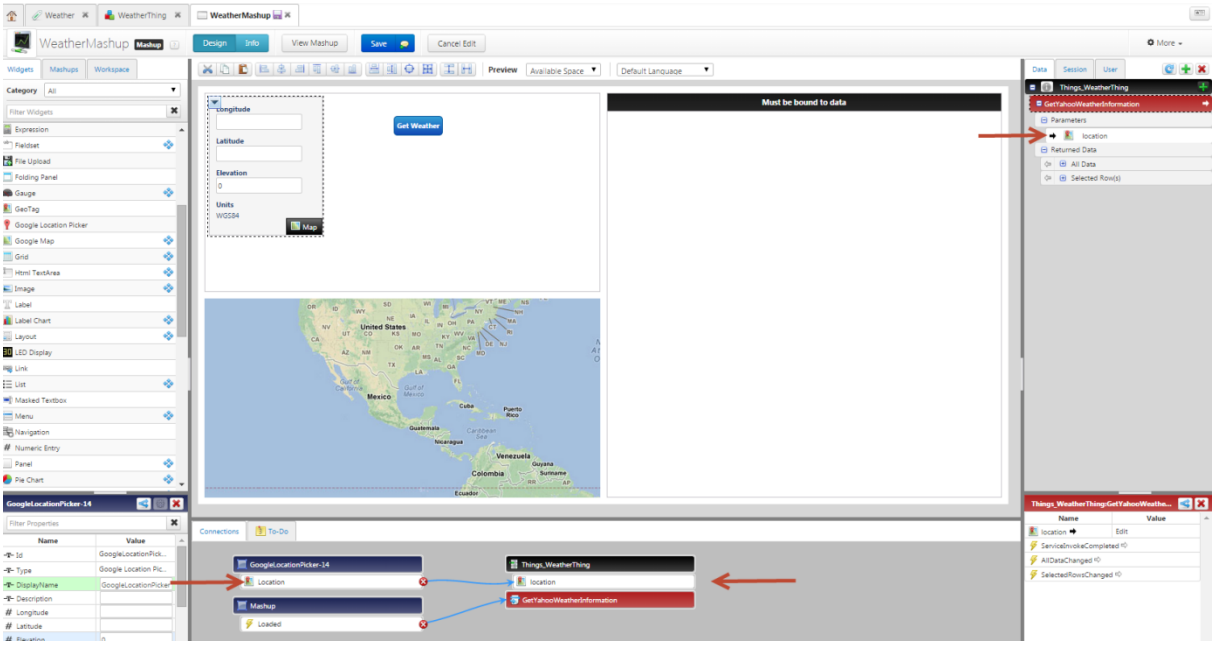
- ***Bind the Google Location Picker Widget*** *– you bind this widget to the location parameter from the GetYahooWeatherInformation service. You do this in order to be able to choose a city (its location) that you need to retrieve weather data information from. You do this by selecting the widget. Next, going over the up left arrow opens a menu from which you select "Location" as shown below:*



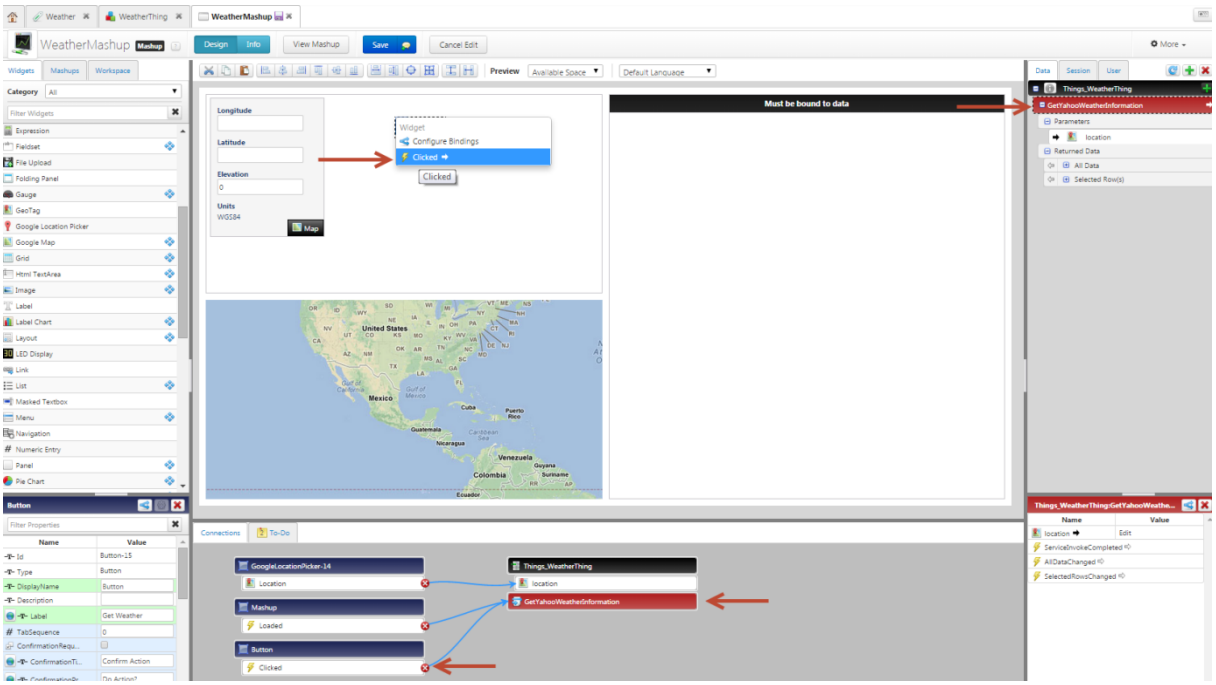
- *Next, drag and drop "Location" to the location parameter in the Service as shown below:*



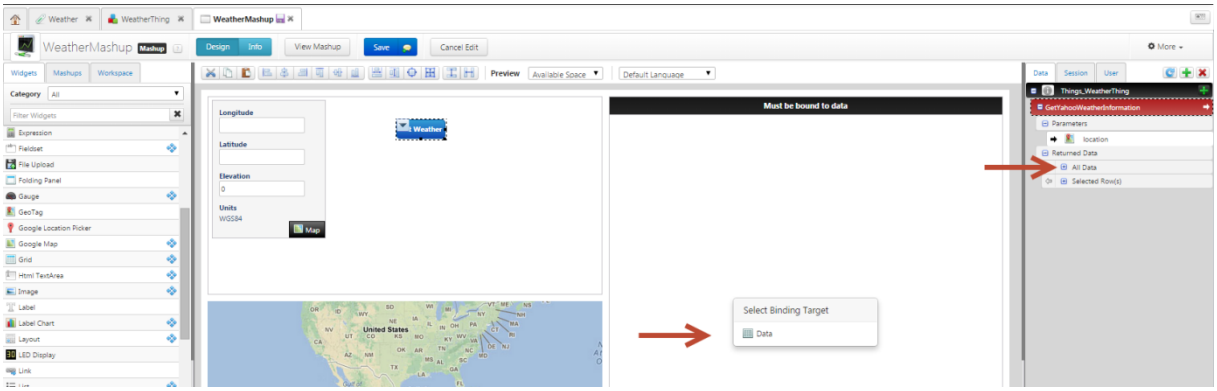
- *Next, If the binding is ok, the arrow from location is black and you can see in the **Connections** tab from the bottom of the mashup the binding as shown below:*



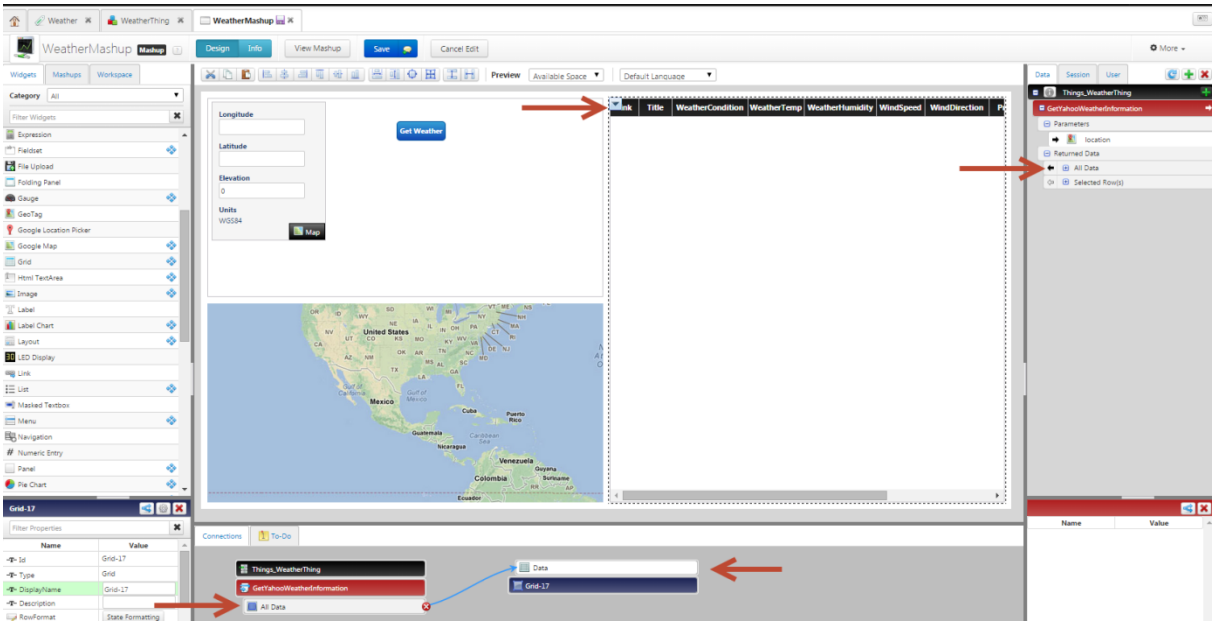
- ***Bind the Button Widget (Get Weather)** – you bind this widget to the GetYahooWeatherInformation service. You do this in order to retrieve yahoo weather data information for the city that you've chosen in the Google Location Picker. Press this button after you choose the City. The purpose of the button widget is to execute the "GetYahooWeatherInformation" service every time it's pressed. You do this by selecting the button widget. Next, going over the up left arrow opens a menu from which you select the **Clicked** value and bind (drag and drop) it to the whole GetYahooWeatherInformation service. The mashup looks like this after the binding:*



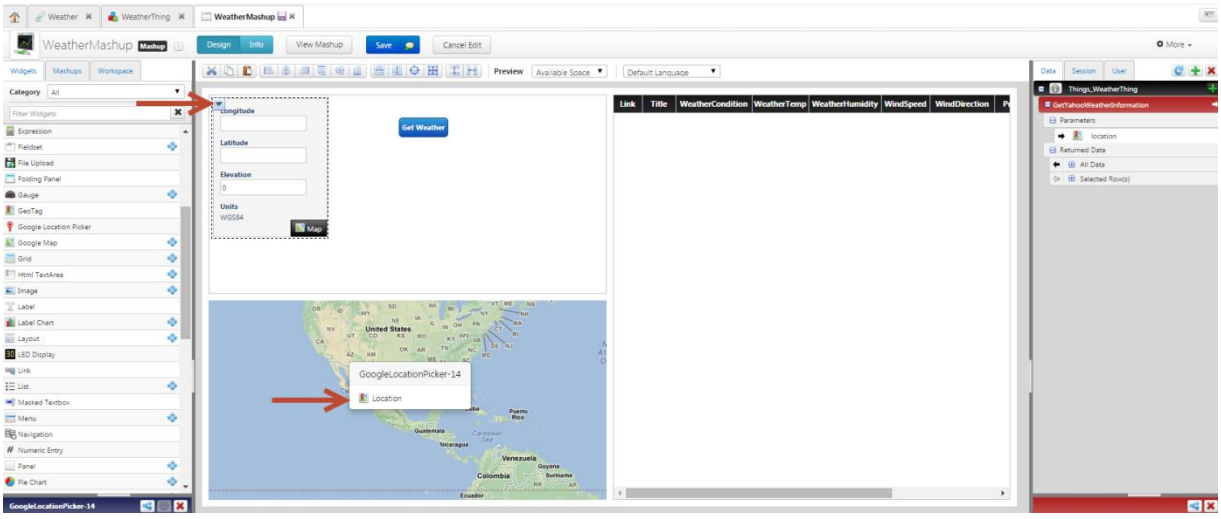
- ***Bind the Grid Widget*** *– you bind this widget to the* ***All data****, underneath the* ***Returned Data****, underneath the* ***GetYahooWeatherInformation*** *service. You do this in order to see the data weather information gathered by the service coming from the Yahoo Weather site. You do this by drag and drop the* ***All data*** *from the service to the grid widget (this time the binding is made the other way around: from the service to the widget) and select* ***Data*** *as shown below*:



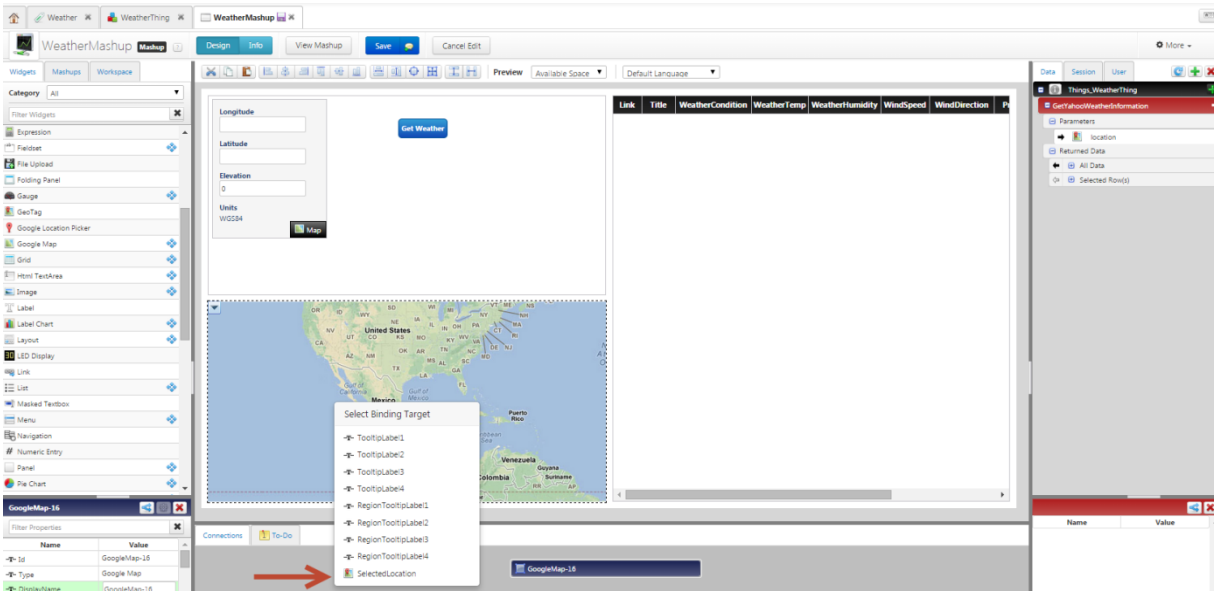
- *If the binding is correct, the arrow from* ***All data*** *is black, the grid has all the fields that you defined in the YahooWeatherFeed Data Shape that you assigned to the GetYahooWeatherInformation service and you can see the binding between the widget and the service in the* ***Connections*** *tab from the bottom of the mashup as shown below:*



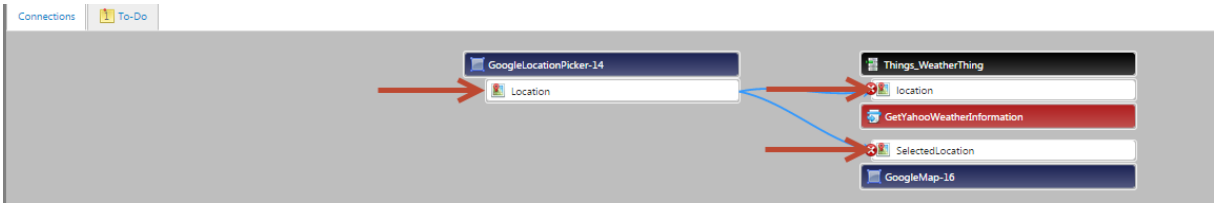
- ***Bind the Google Location Picker widget to the Google Map widget*** *– this time you bind two widgets to each other. You do this in order to see in the Google Map widget, the value (latitude and longitude) you've chosen in the Google Location Picker. So, after you pick a City with the help of the Google Location Picker, a latitude and longitude is displayed in the Google Location Picker, but in the Google Map widget you see where on the map is that City located. You do the binding by selecting the Google Location Picker widget. Next, going over the up left arrow, opens a menu from which you select the "Location" value and bind it to the whole Google Map widget as shown below:*



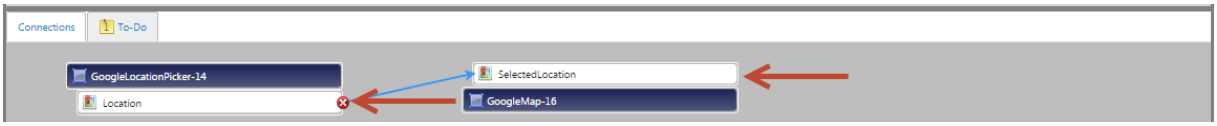
- *Next, when you try to drag and drop the "Location" value from the Google Location Picker, you will have to choose with what data you want to bind it to the Google Map widget. You bind it to the "SelectedLocation" value from the Google Map widget as shown below:*



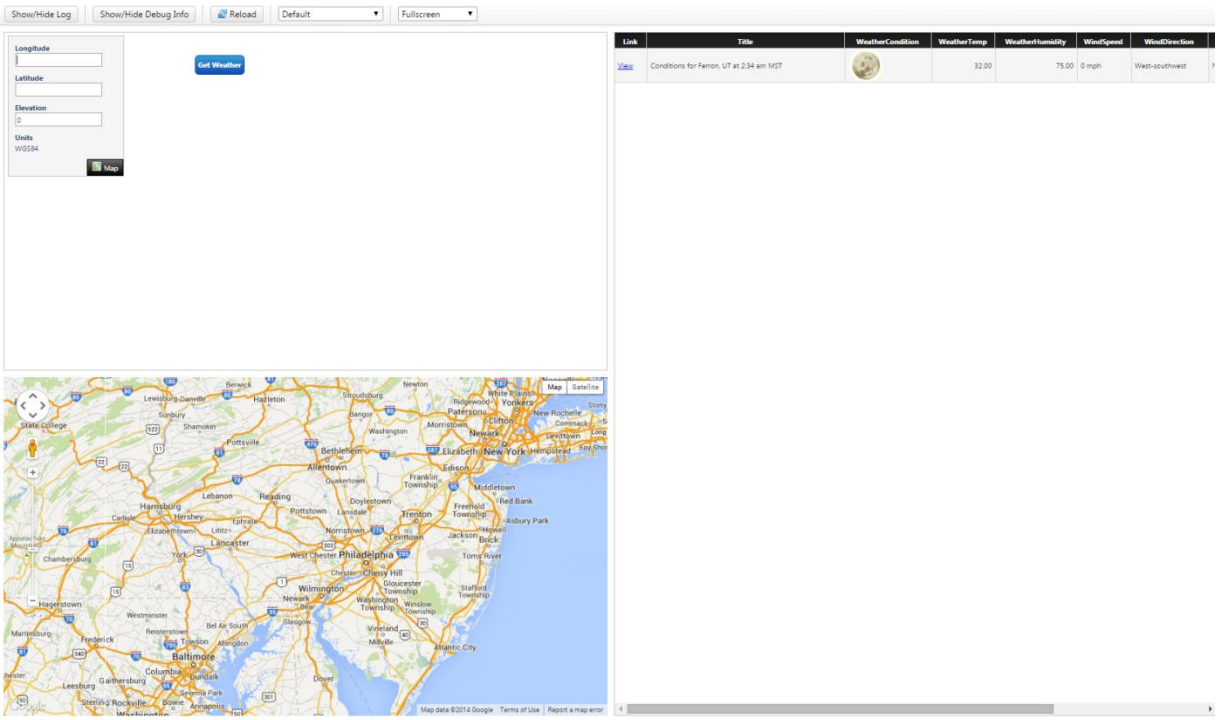
- *If the binding is correct, you see the following information:*
  - *if you choose the Google Location Picker, the **Connections** tab is:*



  - *if you choose the Google Map, the **Connections** tab is:*



- *Now, you need to save the mashup by pressing the **Save** button. Now, you want to view it. Press the **View Mashup**. If that doesn't work (on some browsers it doesn't work), first press the **Cancel Edit** button and then press the **View Mashup** button. The mashup looks like shown below:*



Summary

In the first phase of this Predictive Weather Mashup you have covered some of the most common features of the ThingWorx platform. You first created a simple tag that you used to tag all your mashup's components. Then you've created a simple thing, defined a service (in which you stored weather information) for it and then displayed that service in a mashup.

Once the yahoo thing was created along with his service, you then created the simple mashup that took all of the work you had done and created a visualization that provided you an interactive view for our Yahoo weather data.

In the next phase you will expand upon these concepts:

- You create a value stream, which you use in order to show hystorical and current weather data information in a chart and to export that information;

- You create a new service, which you use in order to show weather data forecast for the next days;