12
Prototyping Using SketchFlow

A prototype is an initial model of a product or application to verify the design and implementation approaches. These prototypes help us in showing the viability of an idea or design. Generally, these prototypes that we create are inexpensive, easy to create, and easy to throw away. These prototypes are also useful to verify that we are moving in the right direction.

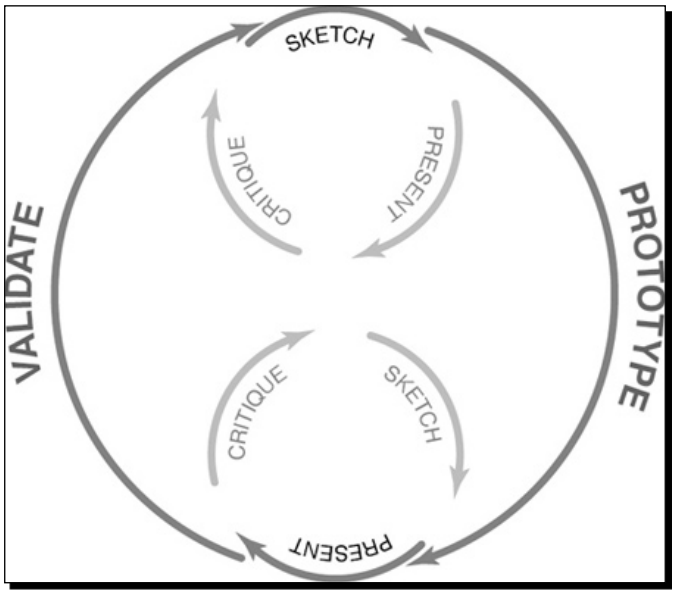
Welcome to the bonus chapter. In this chapter, we will have a look at the following topics:

- Creating SketchFlow
- Packaging SketchFlow
- Sharing SketchFlow
- Adding feedback to SketchFlow
- Application prototyping

What is application prototyping?

Blend provides us with SketchFlow and prototyping. It gives us the capability to create a prototype of an application that is in a raw design style but still functional. This helps us in getting feedback on the core functionality of the application, rather than the view or presentation of the application, without getting distracted from the core features of the application.

This prototyping with SketchFlow helps us follow the prototype design model in which we create a working prototype, get feedback on the design, and then incorporate the feedback to create the next working prototype. This process continues until we reach the phase where we can move from design to implementation. The following is a typical life cycle of a prototype design with SketchFlow:



The two major steps in this process are prototyping and validation. We prototype a part of the application into a working model, and then we validate it by taking feedback from the stakeholders. In our case, this prototyping refers to sketching the application, and validation refers to presenting our idea to the stakeholders and getting their critical feedback to improve the prototype.

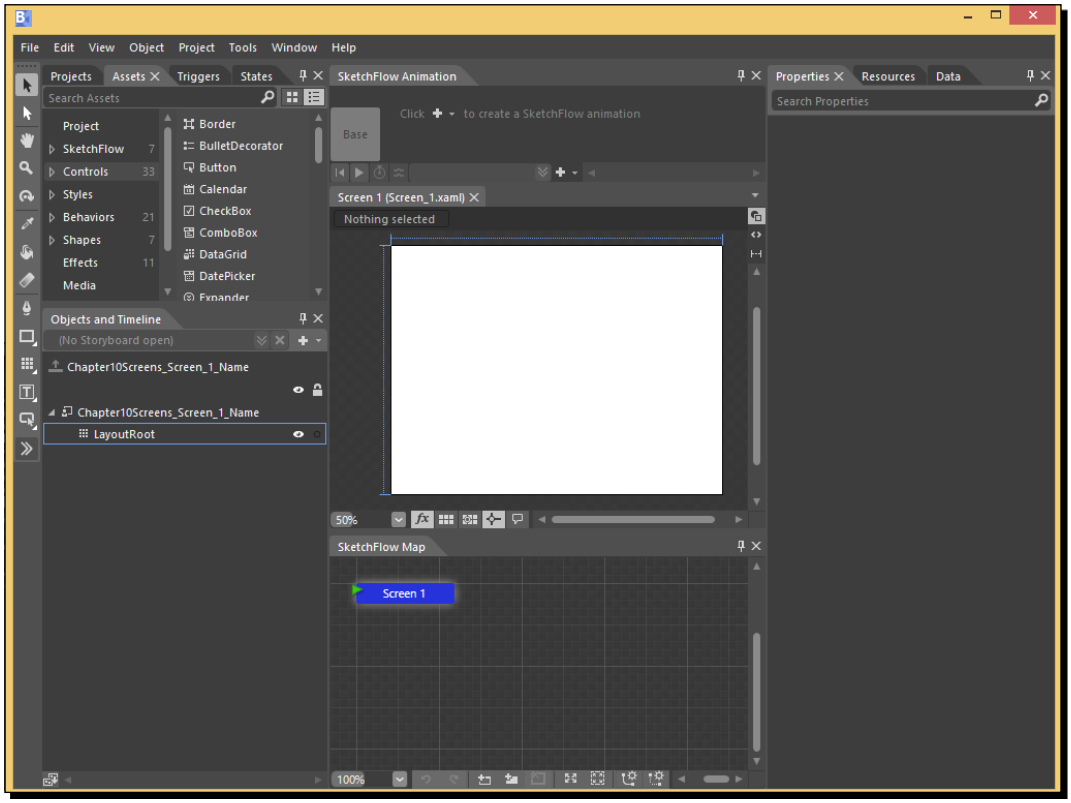
Creating SketchFlow

In this section, we will create a SketchFlow application and have a look at the various components of the SketchFlow application.

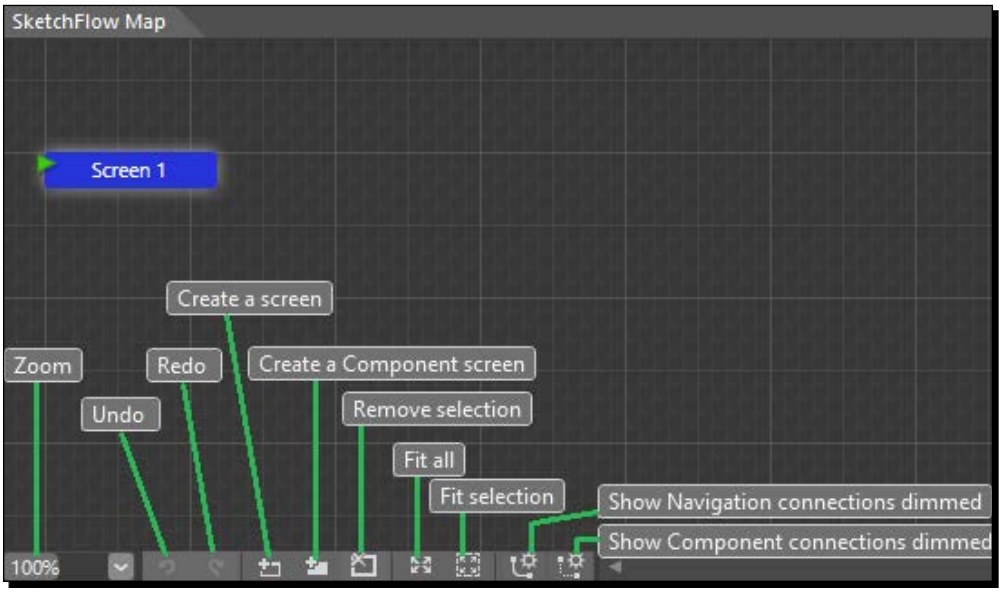
Time for action – creating a SketchFlow application

Here's how we create a SketchFlow application:

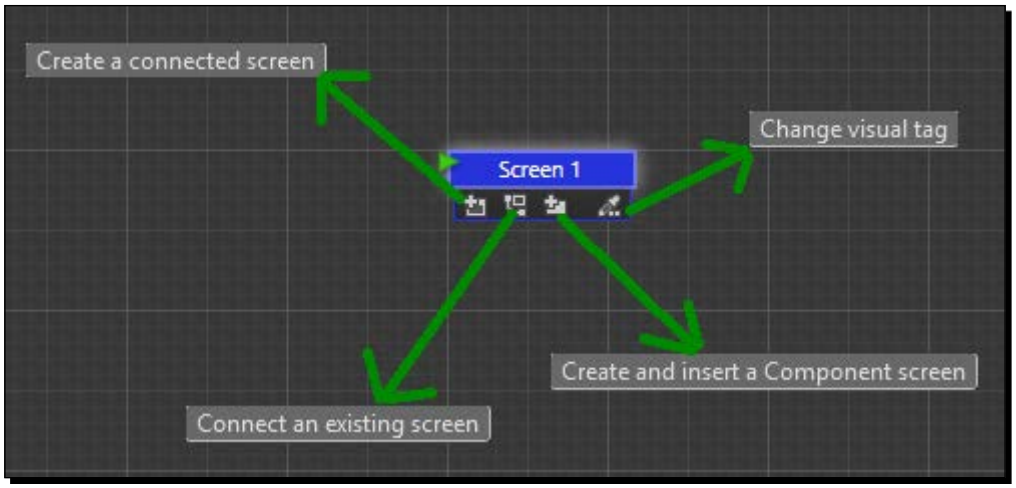
1. Open Blend for Visual Studio 2012 and select **WPF** from the left-hand side panel. Once you do that, you will see options. Select the **WPF SketchFlow application** and name the project `Chapter12`.
2. Once you click on **OK**, you will see that our screen seems similar to the ones we have seen until now, as shown in the following screenshot:



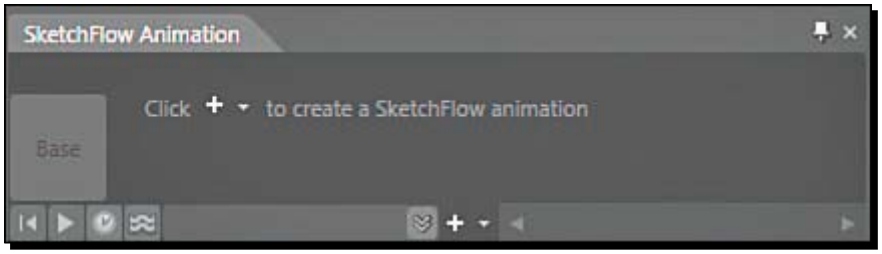
3. The first new panel that you will notice is the SketchFlow Map window at the bottom. If you look closely, you will see a few buttons at the bottom of this panel. These buttons are (starting from the left-hand side) **Zoom**, **Undo**, **Redo**, **Create a screen**, **Create a Component screen**, **Remove selection**, fitting options followed by options to dim navigation and component connections. This is shown in the following screenshot:



4. The most eye-catching part of this screen is **Screen1**, in bright blue, in the top-left corner of the panel. When you hover the mouse over that button, you will see more options (as shown by the green arrows in the following screenshot). These are the options to **Create and insert a Component screen**, **Change visual tag**, **Create a connected screen**, and **Connect an existing screen**.



5. Another panel that we need to analyze is the SketchFlow **Animation** panel. Select the **Animation** panel, and it will look somewhat like the following screenshot. This is available at the top of the art board by default. We will have a look at this panel later in the chapter, when we create SketchFlow animations.



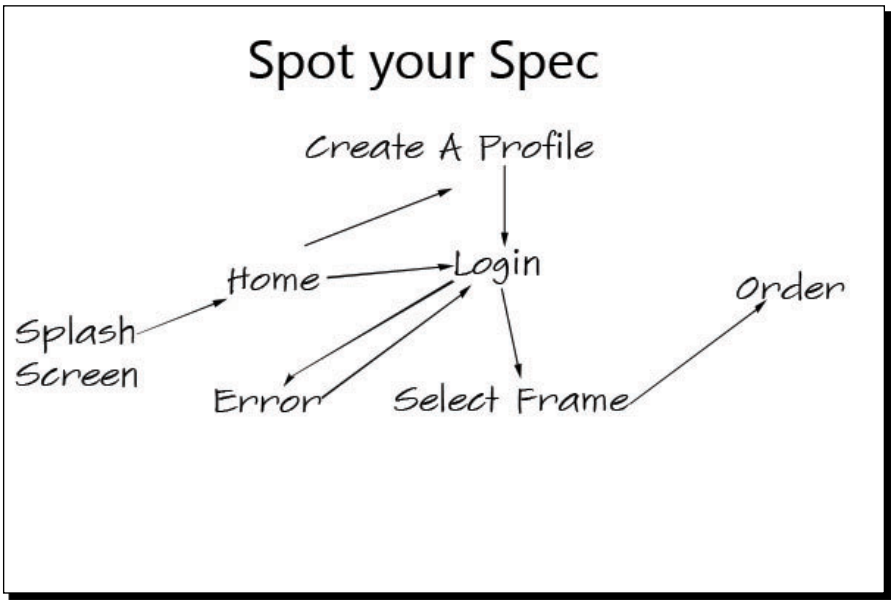
What just happened?

We created a blank SketchFlow application and had a look at the various panels available to design the SketchFlow application.

Now that we have an understanding of the panels available in SketchFlow, we will create a simple working prototype of the application.

Let's suppose we have an application in which we allow the user to select a custom frame for spectacles and types of lenses and get it delivered to their place. There will be a workflow that our application will allow the users to follow to complete their order. The design of the workflow of the application follows shortly.

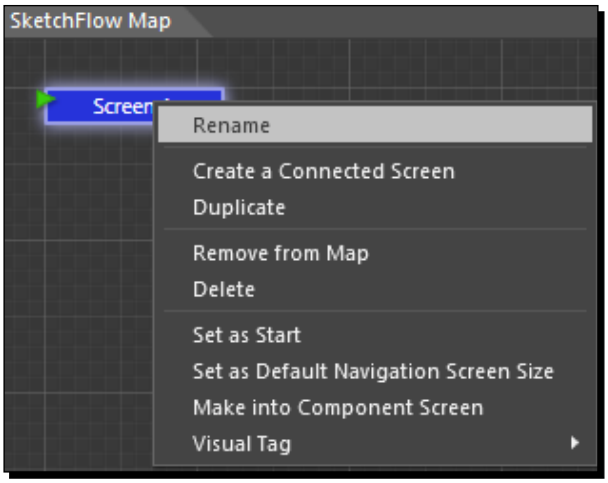
When the application starts, the splash screen comes up, and then the home page loads, which gives the user the option to either log in or create a profile. The user should see an error if the login is unsuccessful, and if the login is successful, then the user can go ahead and select a frame and then the lens they want with it. Then, this choice could be saved, and the order could be placed, followed by payment and shipping.



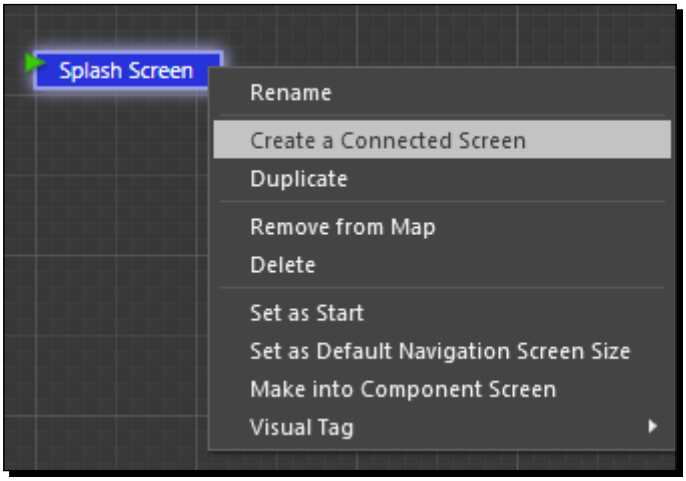
Visuals

Time for action – creating SketchFlow screens

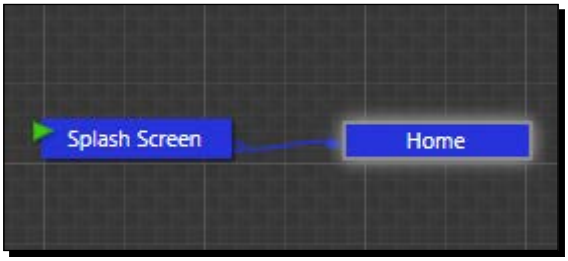
The first thing that we will do is create all the screens mentioned in the preceding design into our SketchFlow map and connect them as described in the screenshot. The next thing that we will do is rename the screen that is available in document map. We can rename the screen by right-clicking on it and selecting **Rename.**



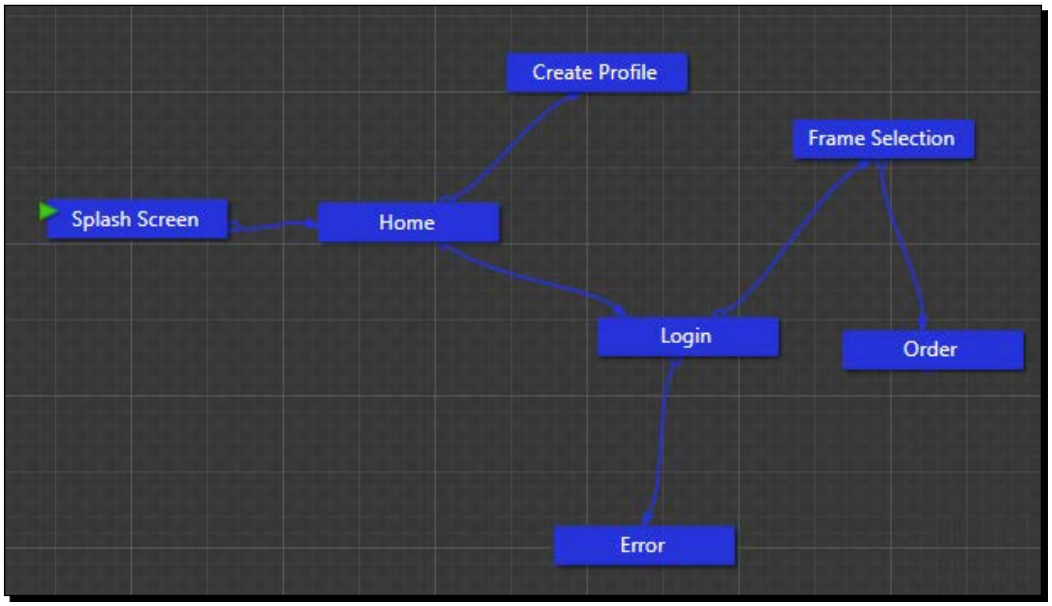
1. Let's rename **Screen1** to `Splash Screen` as this will be the first screen of the application that the user will see when the application is launched.
2. Now, right-click on **Splash Screen** and select **Create a Connected Screen**, as shown in the following screenshot:



3. Once the screen is available, rename it to `Home`, and drag it in front of **Splash Screen**:

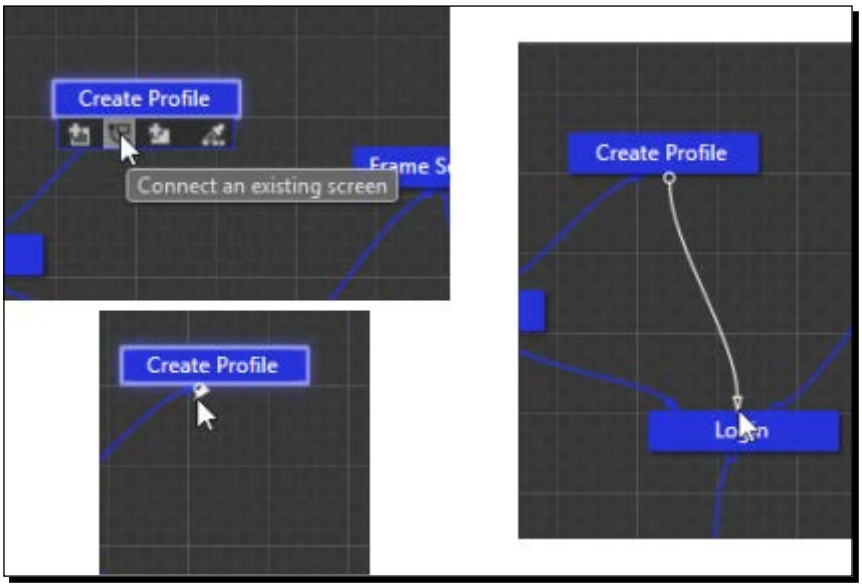


4. Similarly, let's create more connected screens as shown in the following design diagram:

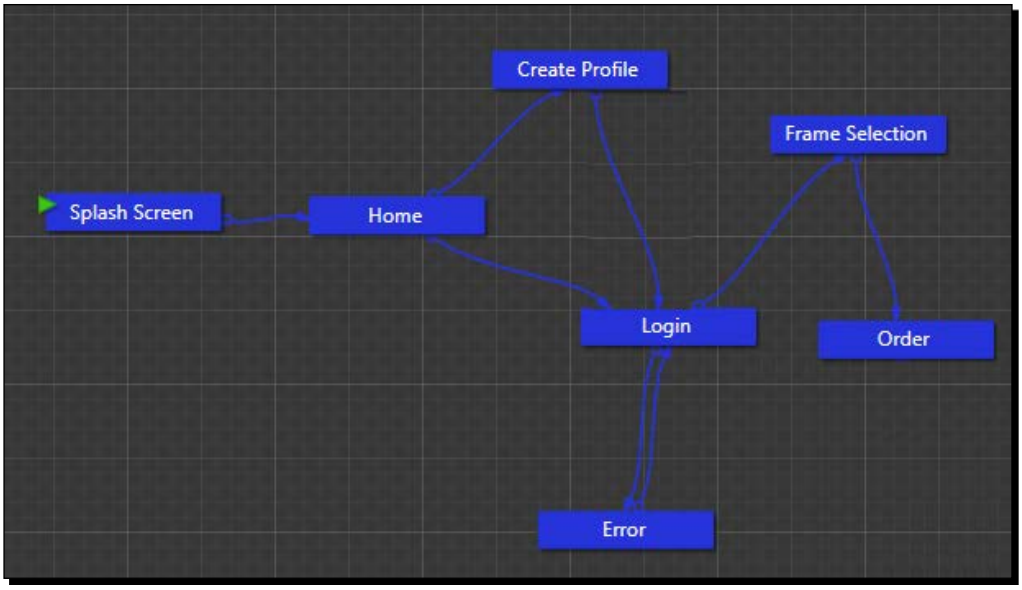


5. We have all the screens available and most of the connections. According to the current design, we have nowhere to go once we reach the **Create Profile** screen or the **Error** screen. So, let's add those connections as well.

6. Mouseover on the **Create Profile** screen and more options come up. Now, click on **Connect an existing screen**, and without lifting the mouse, drag it to the **Login** screen:



7. Similarly, now add a connector starting from the **Error** screen to the **Login** screen and a connecter starting from the **Shipping** screen to the **Home** screen. Then, SketchFlow looks as follows:



What just happened?

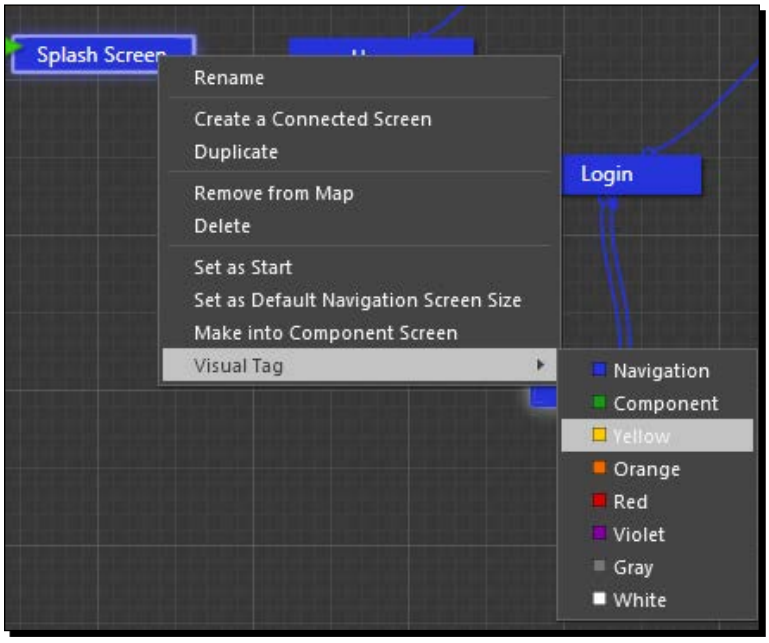
We have a set of screens that are connected, and these together form the SketchFlow application.

Visual states

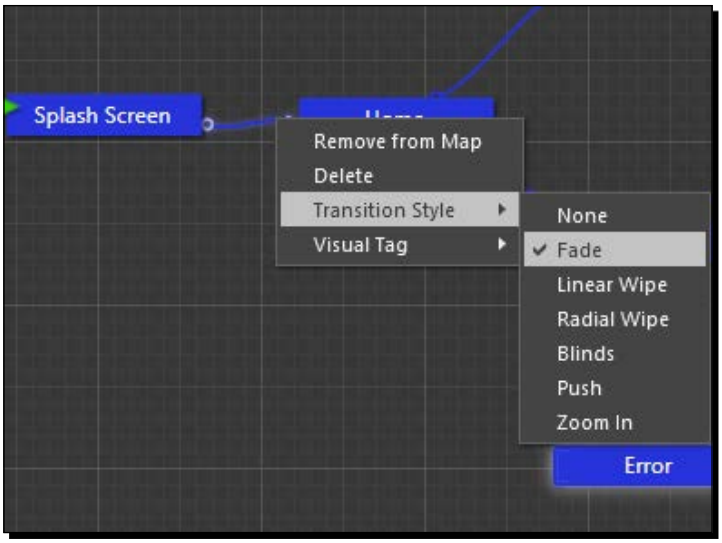
Time for action – color coding SketchFlow screens

Perform the following steps to color code SketchFlow screens:

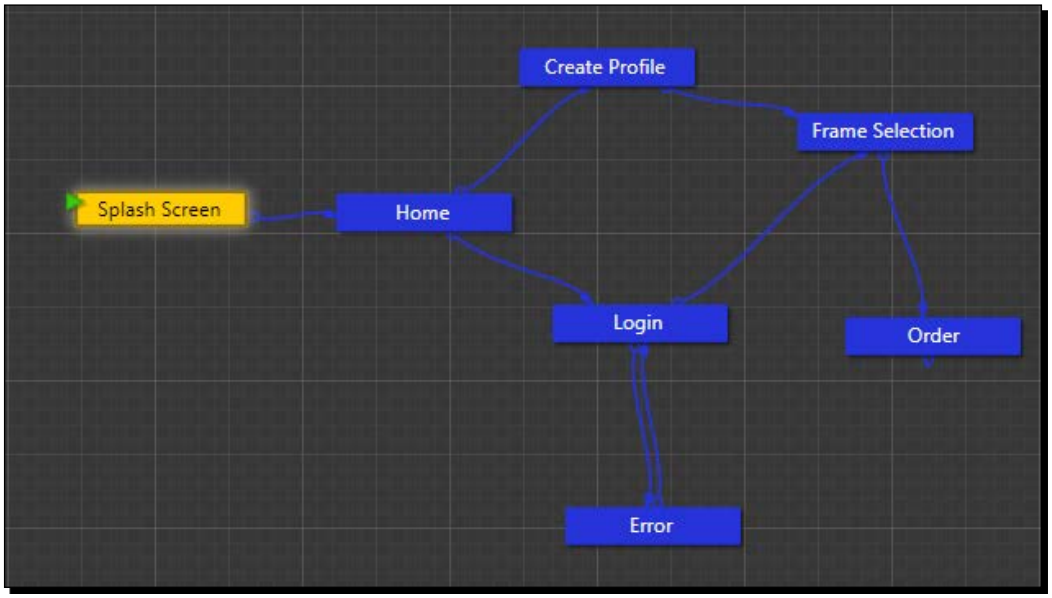
1. This is the start of the SketchFlow application design. The next step is to improve the workflow of the application. We will do this by first changing the colors of the nodes so that we can visually distinguish between the various nodes, as shown here:



2. Along with color coding, we can also add various transition effects to the connectors. This could make our prototype livelier. So, when we run in the SketchFlow player and move from one screen to another, the first screen will fade out and the next screen will become visible:



3. We have given **Splash Screen** the color yellow so that it stands out. Also, you would notice on **Splash Screen** that there is a small green triangle, which symbolizes that it is the start screen:



What just happened?

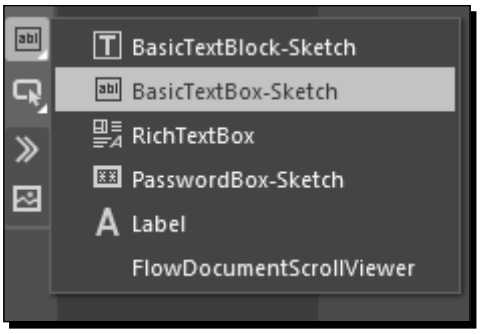
We have color-coded the screens so that we have better visibility of the SketchFlow application.

Design

Time for action – designing a SketchFlow screen

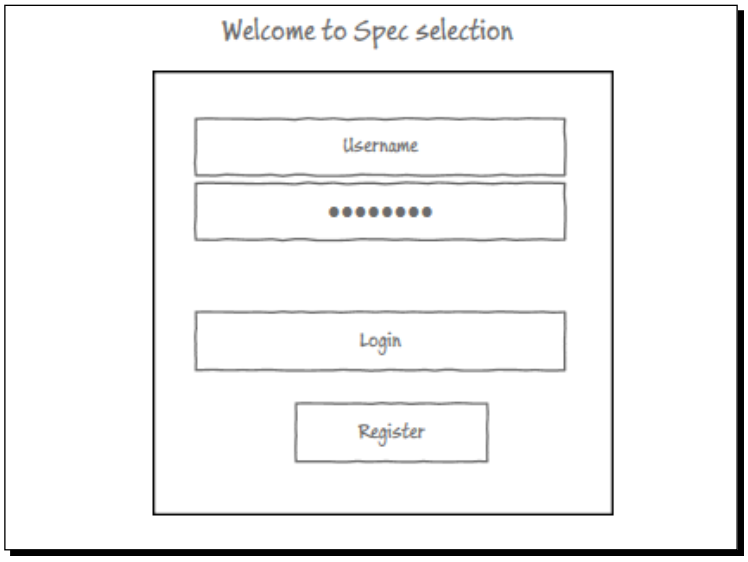
The first screen that we will start from will be the login screen:

1. Double click on Login in SketchFlow map, and `Login.xaml` will open in the art board for us to design.
2. Now, move to the toolbox on the left-hand side, select **BasicTextBox-Sketch** from the text controls, and draw it onto the art board:



3. The TextBox will look as though it's hand drawn, and that's because of the built-in sketch styles. The sketch styles allow us create prototypes without the default, polished-looking controls so that the focus stays on the prototype rather than on the controls.
4. Change the text inside the TextBox to the username by double-clicking on it. The `Login.xaml` file will look similar to the following design diagram:

5. Now, let's add a password box for the password and two buttons for login and registration and put all these controls inside a border. Then, we will also add a TextBlock at the top with the name of the application. Then, we will change the text as shown in the following image:



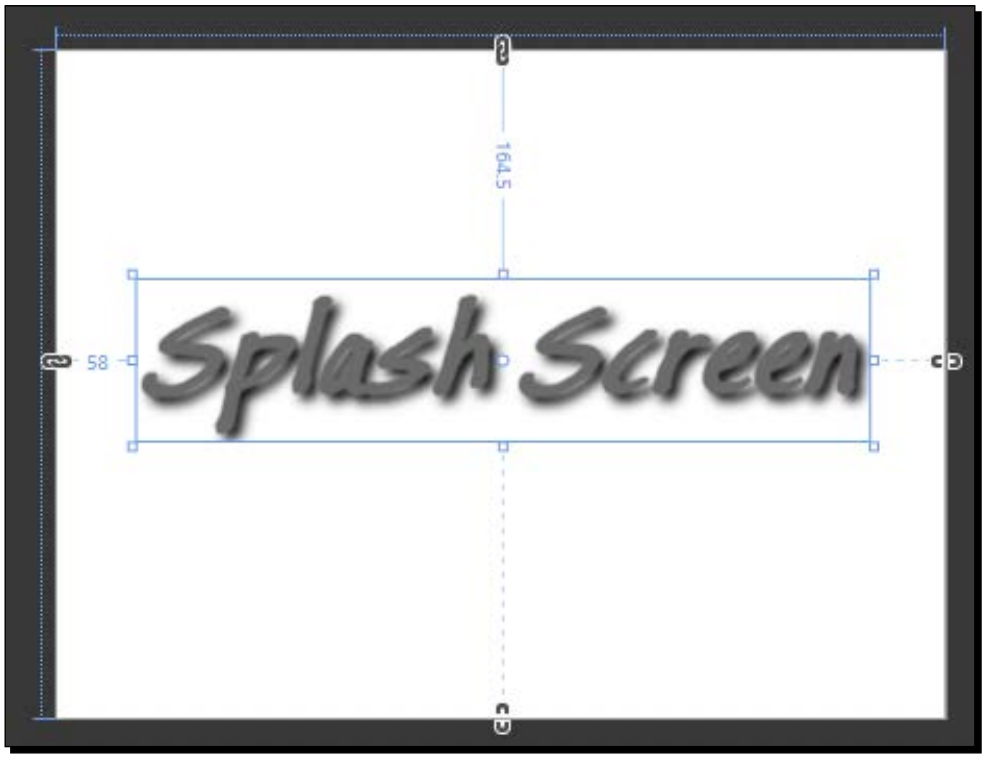
What just happened?

We have designed a screen of the SketchFlow application.

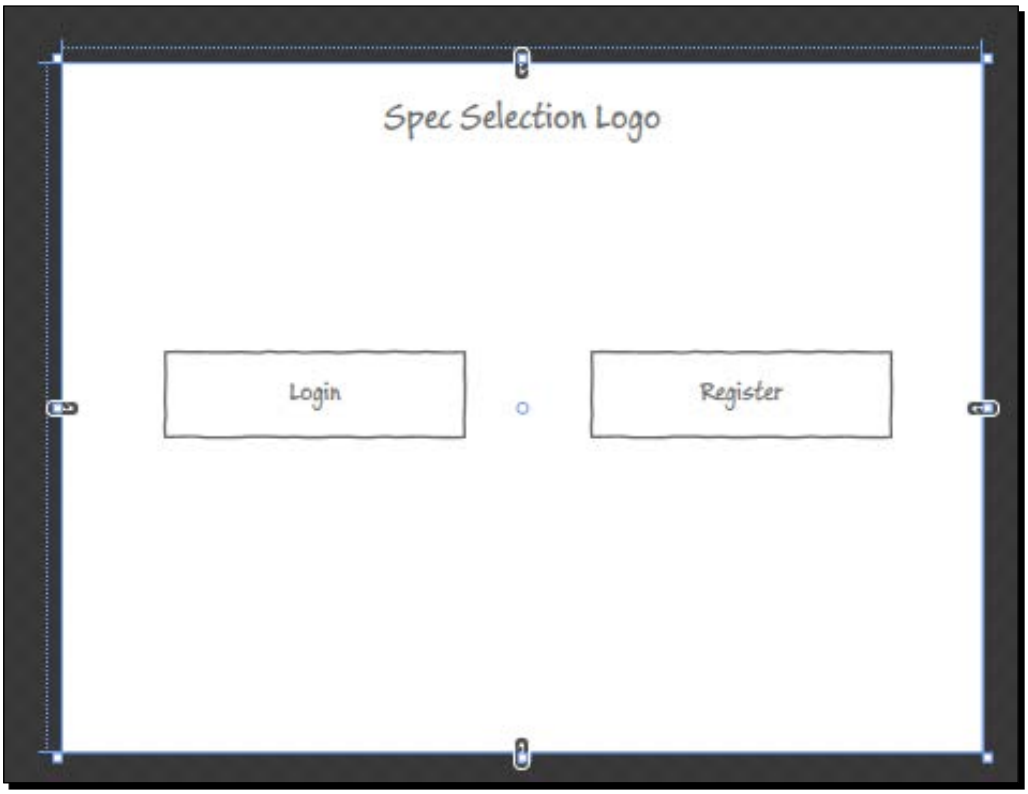
Time for action – designing other screens in the SketchFlow application

Let's now move to splash screen and drop a TextBlock onto it:

1. We will change the text of the TextBlock to **Splash Screen** and increase the font of the text to `72pt`.
2. Select the TextBlock, and, then, from the **Appearance** section in the **Properties** panel, add new **DropShadowEffect**. We will also center the TextBlock so that the text comes to the center of the screen.



3. Now that **Splash Screen** is done, we will move to the **Home** screen, a placeholder for the application logo, and links to navigate to **Login** or **Register** as follows:



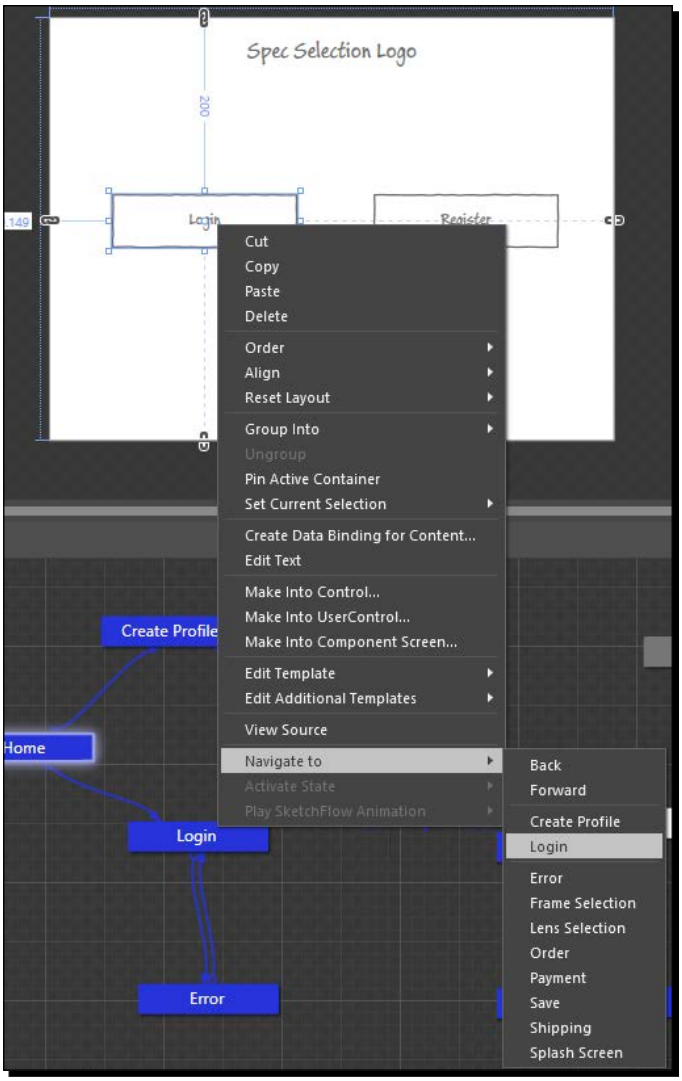
Navigation

Now, it's time to see another cool feature available in Blend. This will help us configure the navigation of the application.

Time for action – adding navigation

To add navigation, perform the following steps:

1. To use this on the **Home** screen, we need to right-click on the **Login** button and select navigate to **Login**. Apart from the **Login** screen we selected, we also see all the other screens we could navigate to. We have the option to go back (this enables the user to navigate to the previous screen in the navigation sequence if any) or forward (this enables the user to navigate to the next screen in the navigation sequence if any) as well:



2. Do the same thing with the **Register** button, but, this time, we will select **Create Profile**.

What just happened?

We have added navigation to the screens in our SketchFlow application.

SketchFlow Player

Now go ahead and press *F5* to run the application, and we will see SketchFlow Player showing our application:

- The first thing that we will notice is that the center area is occupied by splash screen. This is the area where the screens will change as we navigate through the application.
- The top section of the left panel has a few interesting options:
  - There are two icons present next to the SketchFlow Player label. These will allow us to float or collapse the left-hand side panel.
  - The next set of options that we see are as follows:

    Home button: This takes us back to the first screen.

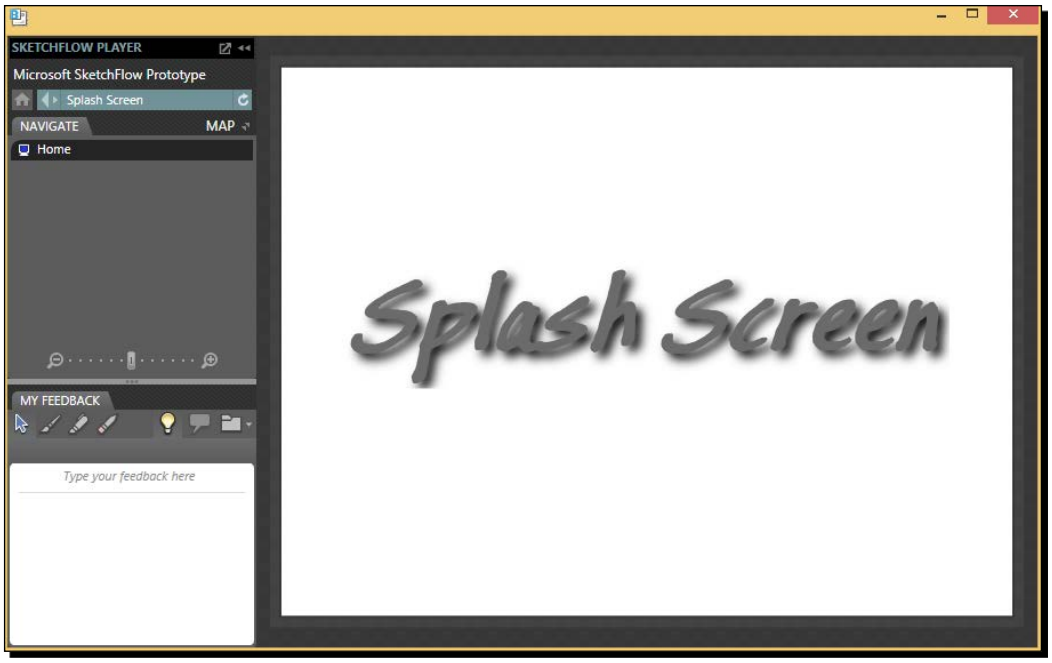
    Back and forward buttons: Using these, we can navigate back and forth in the application.

    Splash screen: This shows the name of the current screen.

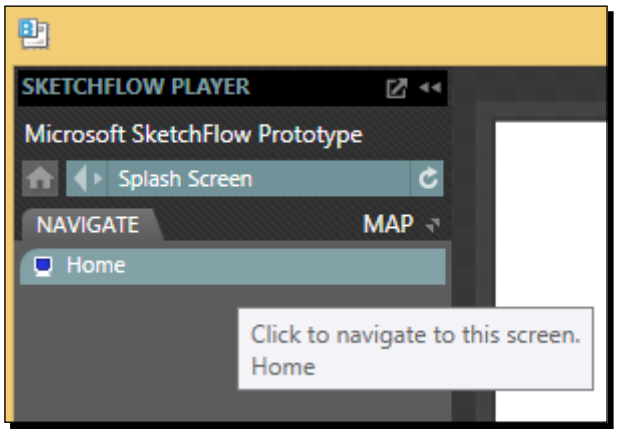
    Refresh: This refreshes the screen to reflect the changes that were made since we came to this screen.

  - We see double triangle icons next to the **MAP** label; these toggle the visibility of SketchFlow map in the center area.
  - Just below **NAVIGATE**, we can see the names of the screens that are next in the navigation chain. Currently, it shows only one option, that of the **Home** screen, as that is the only navigation available as per SketchFlow map.
  - The next set of buttons that we see help us in zooming in and out of the screen

- The bottom-left section of the screen gives us a lot of options to add feedback comments, as shown in the following screenshot. We will look into this in detail later in the chapter.

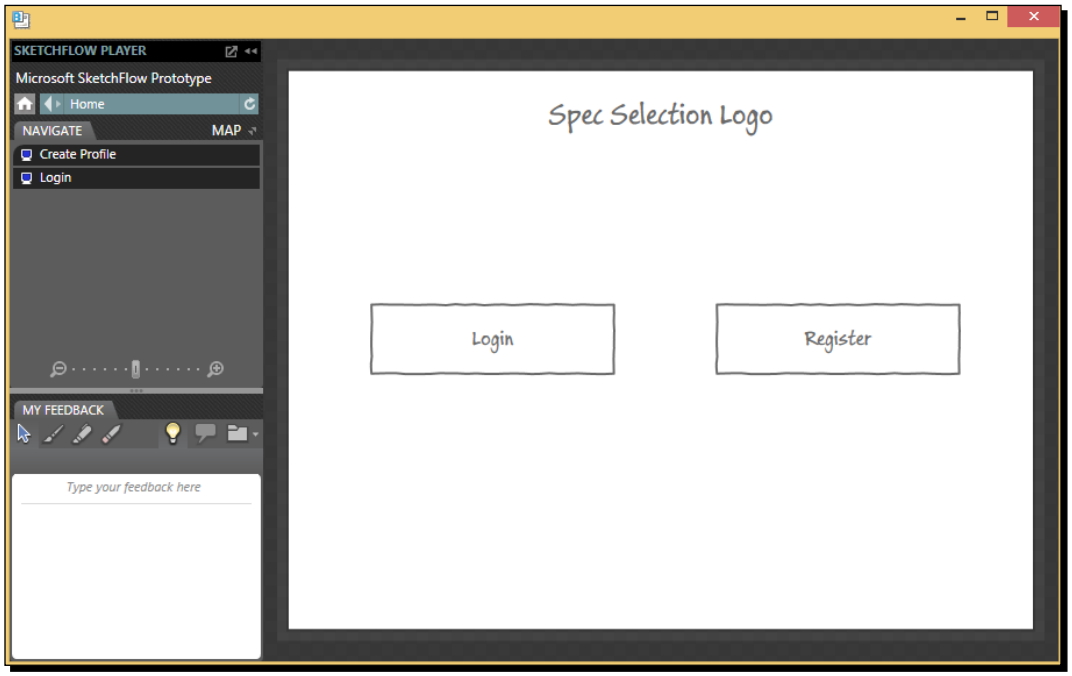


- Click on the **Home** screen and we will move to the next screen as per our SketchFlow map:

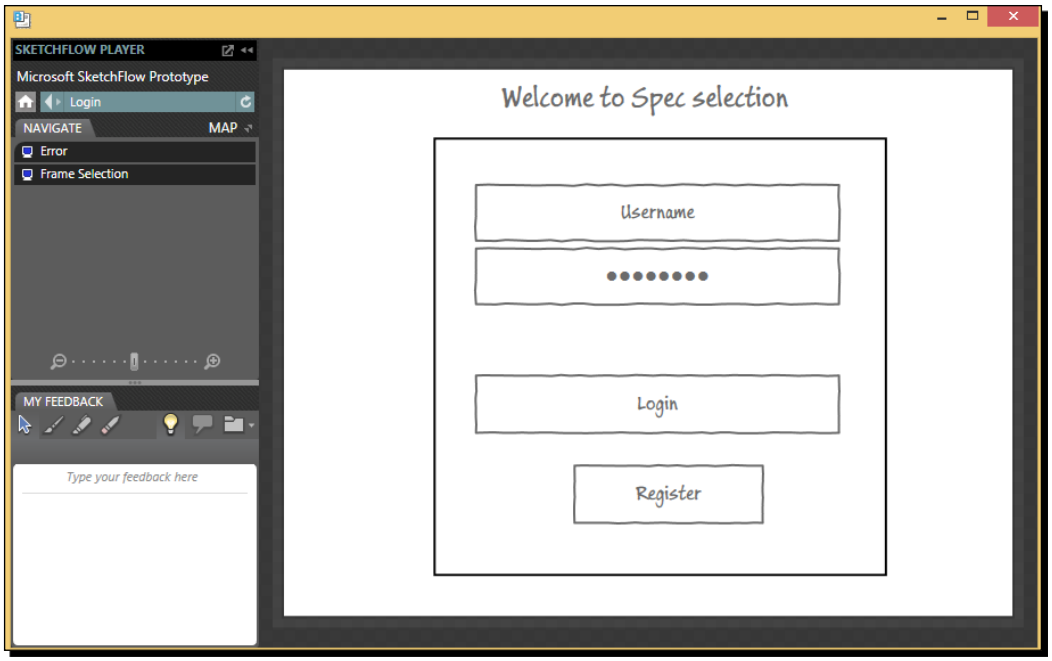


We can see that we have navigated to the **Home** screen and we see two changes, as shown in the screenshot after this list:

- The center area of SketchFlow Player changes to the **Home** screen.
- The left-hand side panel of SketchFlow Player now has different available navigation options:



- Now, let's click on the **Login** button on our **Home** screen, and we will navigate to the login page since we configured the **Login** button to take us to the login screen and the **Register** button to take us to the **Create Profile** screen:



Getting feedback in SketchFlow

Now, it's time to have a look at the feedback capabilities available in SketchFlow.

There are three ways to provide feedback:

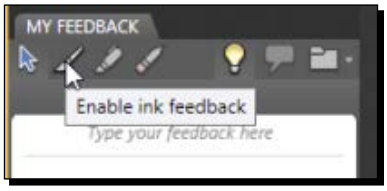
- **Ink**: To give feedback using ink, we can use the pen tool available in the feedback panel.
- **Highlight**: To give feedback using highlight, we need to select the highlighter available in the feedback panel.
- **Text**: To give feedback in text, we just need to add notes in the feedback panel.

We will have a look at all the ways one by one.

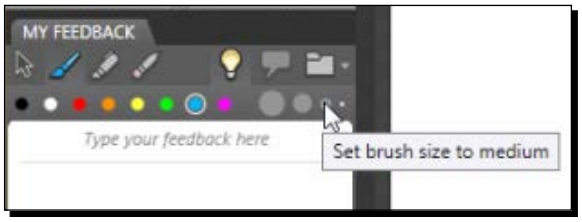
Time for action – adding feedback in SketchFlow applications with ink

Perform the following steps to add feedback in SketchFlow applications with ink:

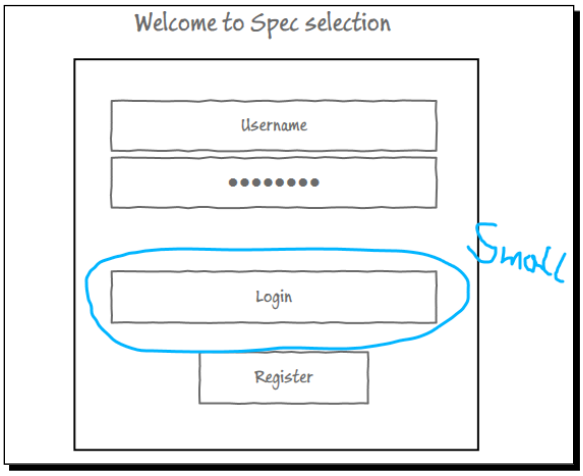
1. In SketchFlow Player, move to the feedback panel and click on **Enable ink feedback**.



2. Once we do that, we will see that we have options to select the color of the brush and the size of the brush. So, we will go ahead and set the brush color to blue and the brush size to medium, as shown in the following screenshot:



3. Now that we have the feedback mode as ink, we will go ahead and make markings on the screen and write comments. So, we circle the **login** button with blue ink and leave a comment to make that button small:



What just happened?

We have given feedback for the application using ink.

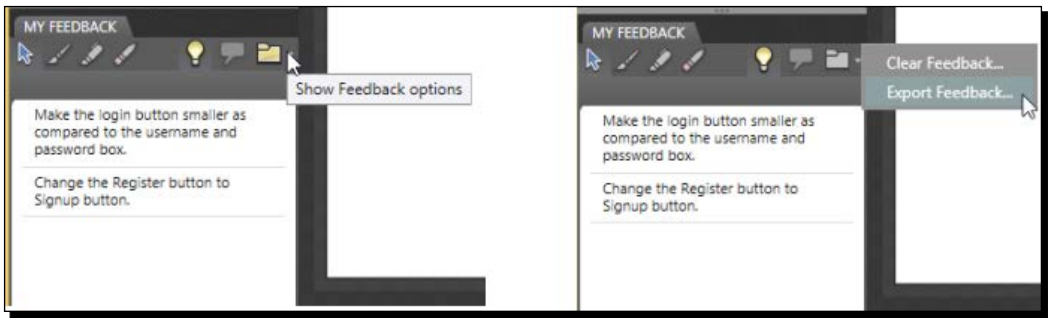
Exporting feedback from SketchFlow

We can also export feedback from SketchFlow.

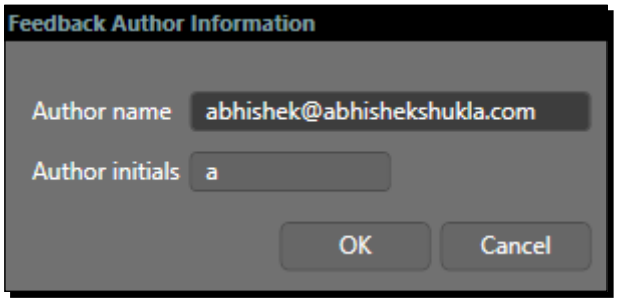
Time for action – exporting feedback from the SketchFlow application

To export feedback from the SketchFlow application, perform the following steps:

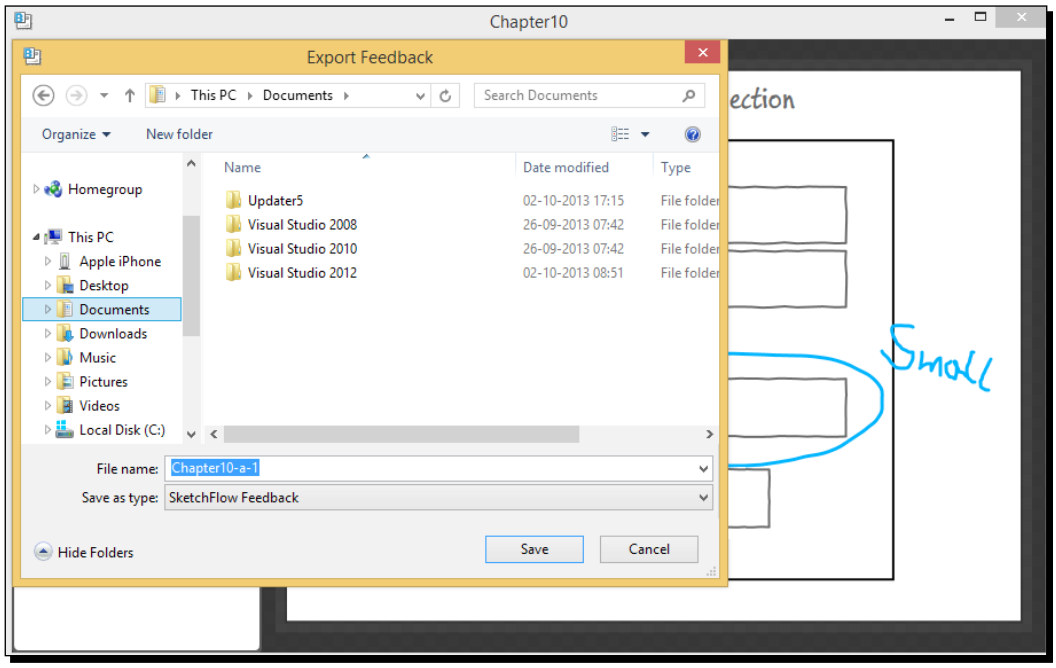
1. Once we click on the **Show Feedback options** icon, we see the following two options as shown in the screenshot after these points:
   - **Clear Feedback…**: This option will clear all the feedback that we have added until now on this screen
   - **Export Feedback…**: This option will allow us to export the feedback.



2. When we click on **Export Feedback…**, we get an option to add our e-mail and initial along with the comment:



3. Once we click on **OK**, we will be asked to save the feedback at a location that can be shared later on, as shown in the following screenshot:



What just happened?

We have exported the feedback that we gave into a file.

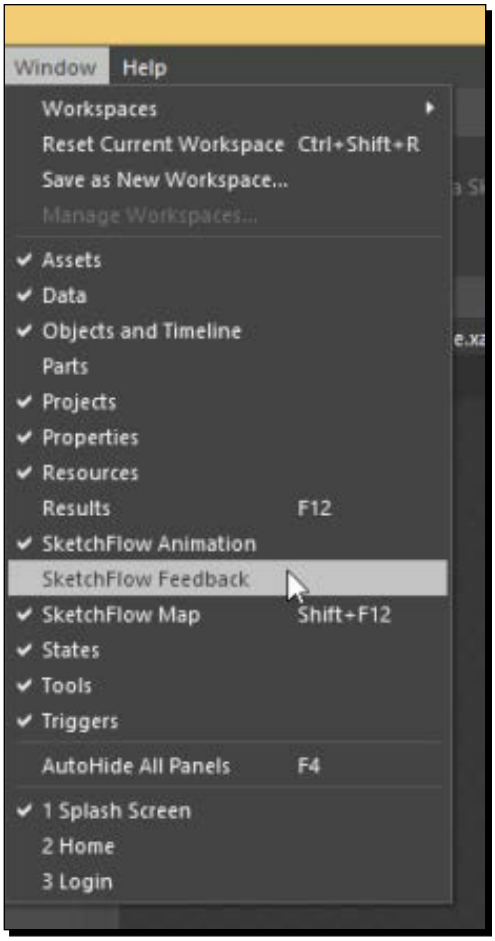
Importing feedback

Now that we have the feedback in the file, it will be really easy to share this file. We can send just this file back on an e-mail and reuse the file. All that the developer or designer needs to do is include it in their SketchFlow project, and they will be able to see the feedback.

Time for action – importing feedback into the SketchFlow application

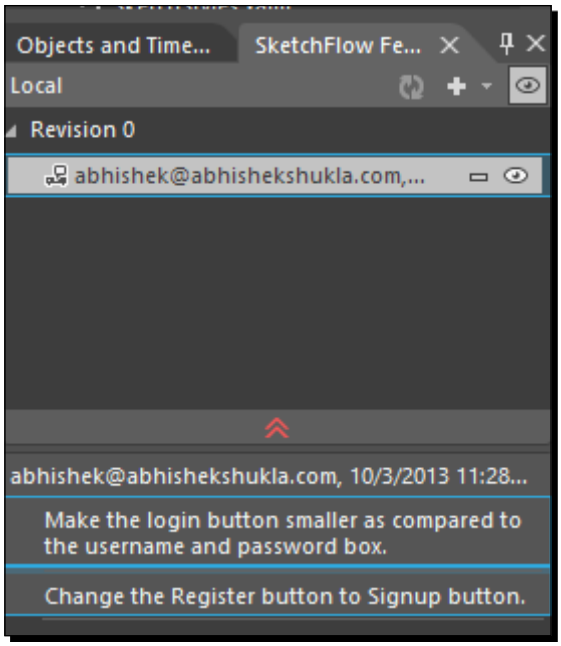
To import feedback into the SketchFlow application, perform the following steps:

1. To import the feedback into the SketchFlow application, we will need the **Feedback** panel, which can be made visible by navigating to **Window** | **Feedback** window.

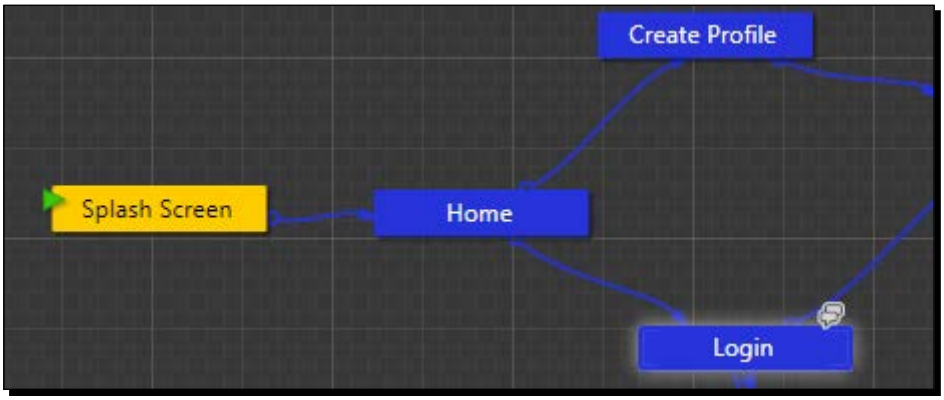


2. Once we see the feedback window, we need to click on the **+** sign to import the feedback file into the application. On doing that, we will see a few things.

3. We will see that the e-mail of the person who gave the feedback is available in the **SketchFlow** panel, and, on selecting that feedback, we will see the comments that were added to that feedback, as shown in the following screenshot:



4. We will also see in the following SketchFlow map above **Login** that there are two small chat boxes above **Login**; these symbolize the two comments that we added to this `Login.xaml` file.



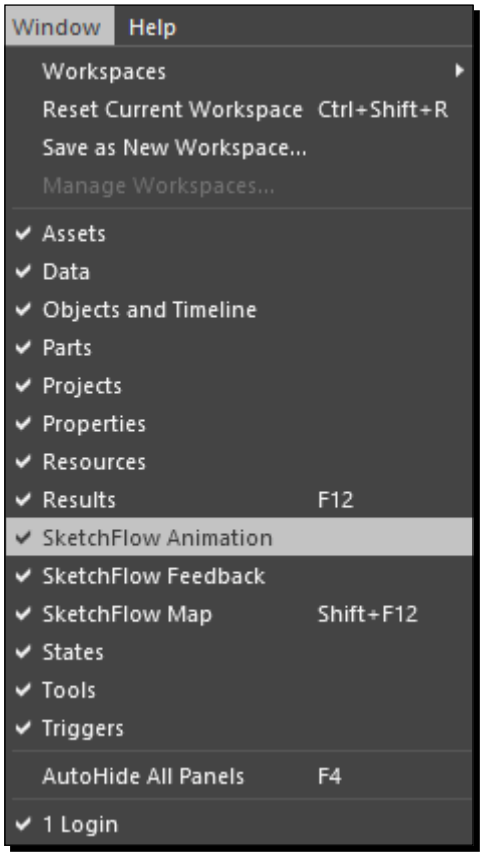
What just happened?

We just imported the feedback from a file. This cycle of feedback and improvement can continue until we have reached a stable prototype.

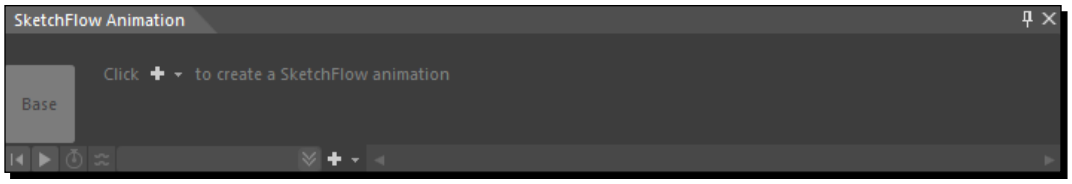
SketchFlow Animation

SketchFlow provides special animation capabilities called **SketchFlow Animation**. We could quickly and easily create animations with advanced interactivity. **SketchFlow Animation** uses Visual State Manager, which we discussed in *Chapter 5*, *Behaviors and States in Blend*.

Let's analyze the **SketchFlow Animation** panel as this is the one we will use in this section. You can open the **SketchFlow Animation** panel from menu by selecting **Window** | **SketchFlow Animation**.



In the **SketchFlow Animation** panel, we see an item labeled **Base**, as shown in the following screenshot. Base is default view of all the elements and their settings and serves as the starting point before the beginning of the animation. Every screen in SketchFlow has a base condition.

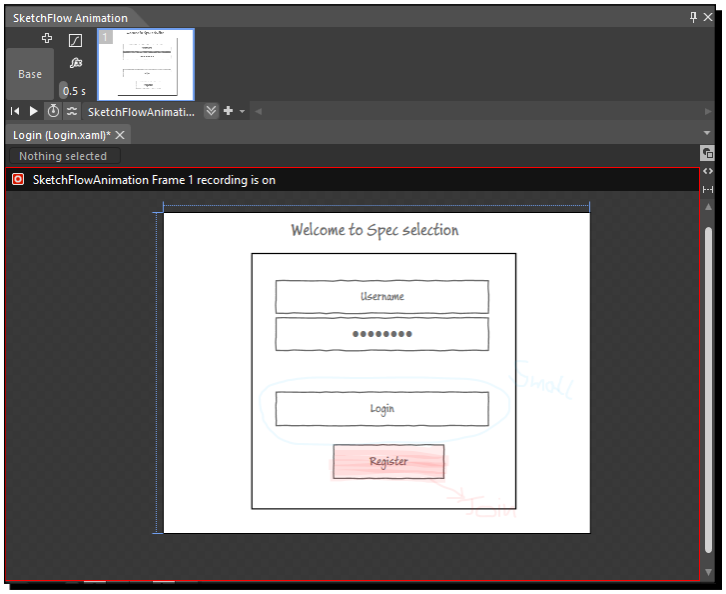


Next to **Base**, we see the **+** control that we use to create animation in SketchFlow.

Time for action – adding a SketchFlow animation

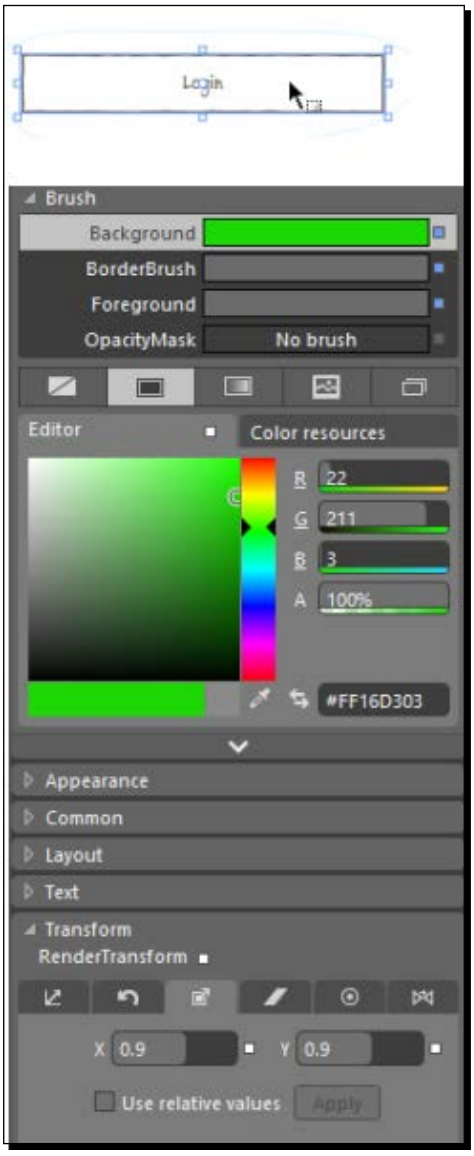
To add a SketchFlow animation, perform the following steps:

1. Let's click on **+**, and we will see a small icon appearing that is the snapshot of the art board but small in size. It is called "frame". We can create these frames by copying the base screen or any other frame.
2. These frames are now the states of our animation. We can record the changes and interactivity to the screen in a frame. We use the term "frame" in animations in SketchFlow, but elsewhere we use the term "states". Both frames and states do the same thing, but the interface in which we animate the items is different. This is shown in the following screenshot:

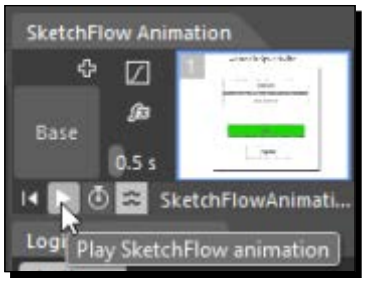


- We would also see the red outline around the art board indicating that the SketchFlow recording is on. This means that whatever we do in the frame will be captured and recorded for the frame we are working on.

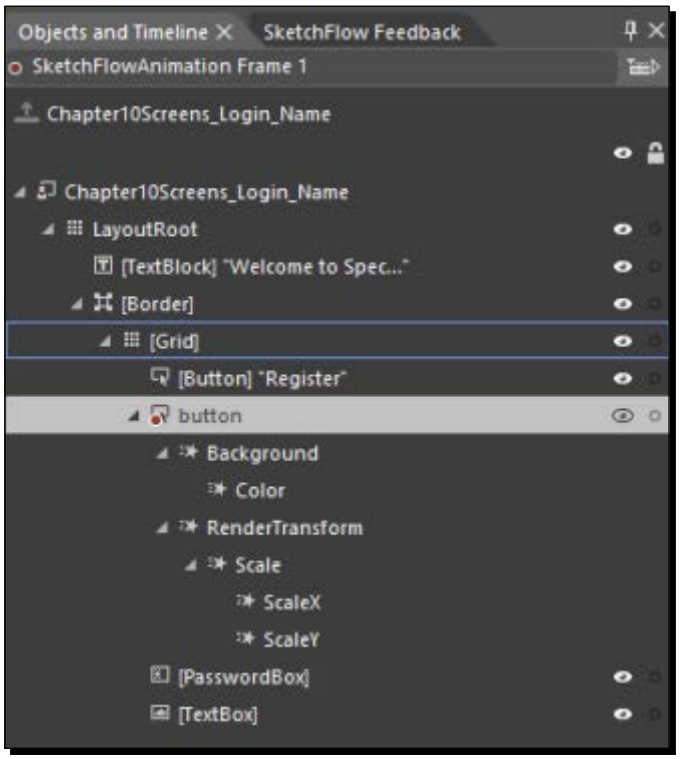
3. Select the **Login** button, and, from the **Properties** panel, give it a green background. Then in the **Transform** section, set a scale transform by setting **X** and **Y** to `0.9`. This is shown in the following screenshot:



4. Now, when you click on the play button in the **SketchFlow Animation** panel, we will see the animation that shrinks the **Login** button to 90 percent of its size and changes its background to green:



5. When we move to the **Objects and Timeline** panel, we see a red dot next to the element that we animated. When we expand the element, we see the properties we have animated, which include background color and the scale transform.



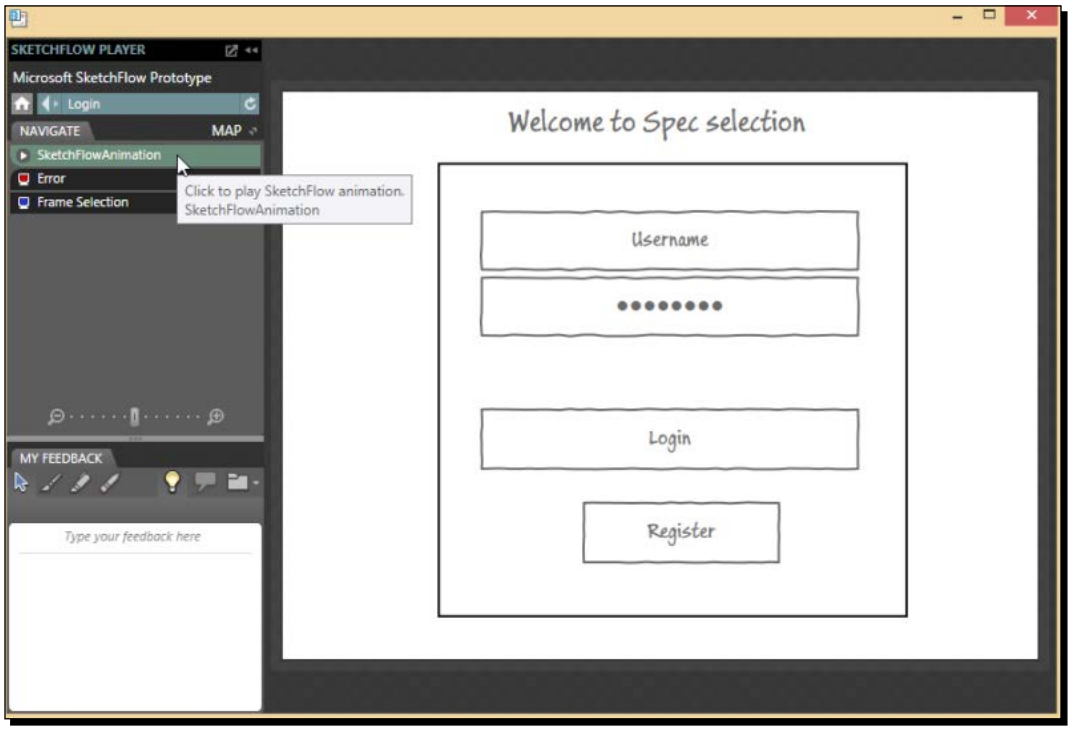
What just happened?

We just created a **SketchFlow Animation**. **SketchFlow Animation** is an easy way to build and demonstrate complex interactions without any dedicated resources, scripting, or code. It enables us to get feedback quickly on the interactions and show the intent iteratively.

Time for action – running a SketchFlow animation

To run a SketchFlow animation, perform the following step:

1. Let's build and run the project, and we will see SketchFlow Player. To run the animation, we will have to navigate to the screen to which we added the animation. Then, click on **SketchFlowAnimation** to play the animation. The following screenshot shows this:



What just happened?

We played the animation in SketchFlow Player.

Animation using states

States allow us to add simple and advanced interactivity to SketchFlow. We had a look at states and Visual State Manager in detail in *Chapter 5*, *Behaviors and States in Blend*. They work in the same way in SketchFlow as well. Using states, we can alter one or more values of an object. Most of the objects come with predefined states, but it is fairly straightforward to customize existing states or define new states for any object.

As discussed in the previous section, every screen has a base condition. This base condition could also be considered to be stateless. When we add one or more states, we enter into a different condition for the screen, where we can change one or more properties. We can enable SketchFlow to transition between these states based on the interactions.

When we modify or create a state (using the **SketchFlow Animation** panel or the **States** panel), we create a new state for the screen that includes every object on the screen. SketchFlow animates the objects to the properties that we have set for those objects in that particular state.

One of the suggested best practices that you could follow is to always create a state that is a copy of the **Base** screen as it would work as a default state to transition back from other states.

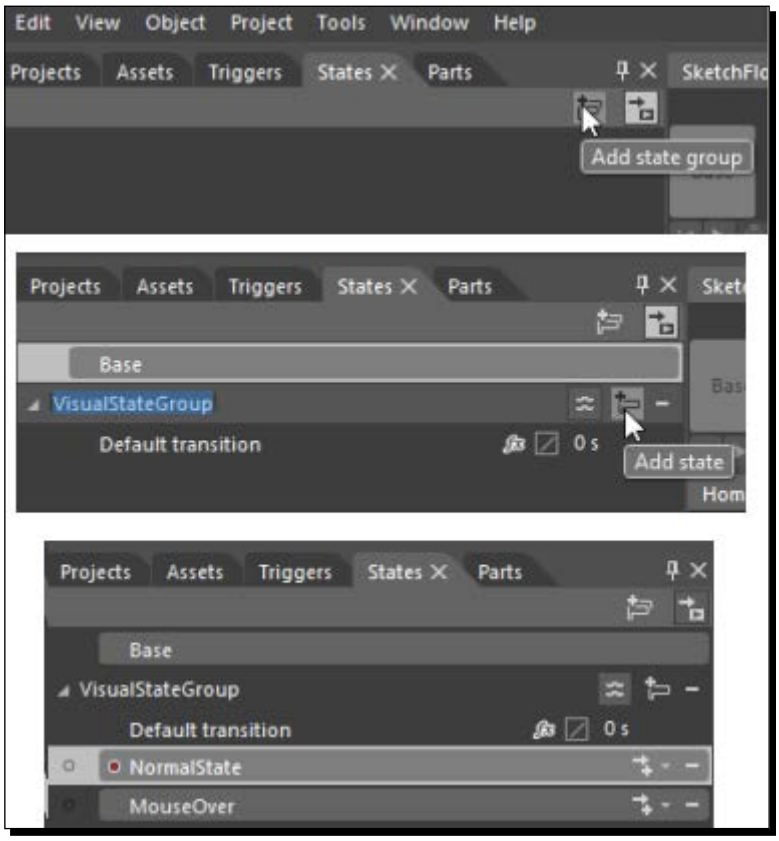
We can accomplish the same things using states and **SketchFlow Animation**. However, **SketchFlow Animation** is generally quicker in places where we want to use linear exposition and don't want to showcase the nonlinear interactions. States could be used to prototype the nonlinear interactivity of the different states of a login page in the event of successful or failed logins.

Time for action – animation using states

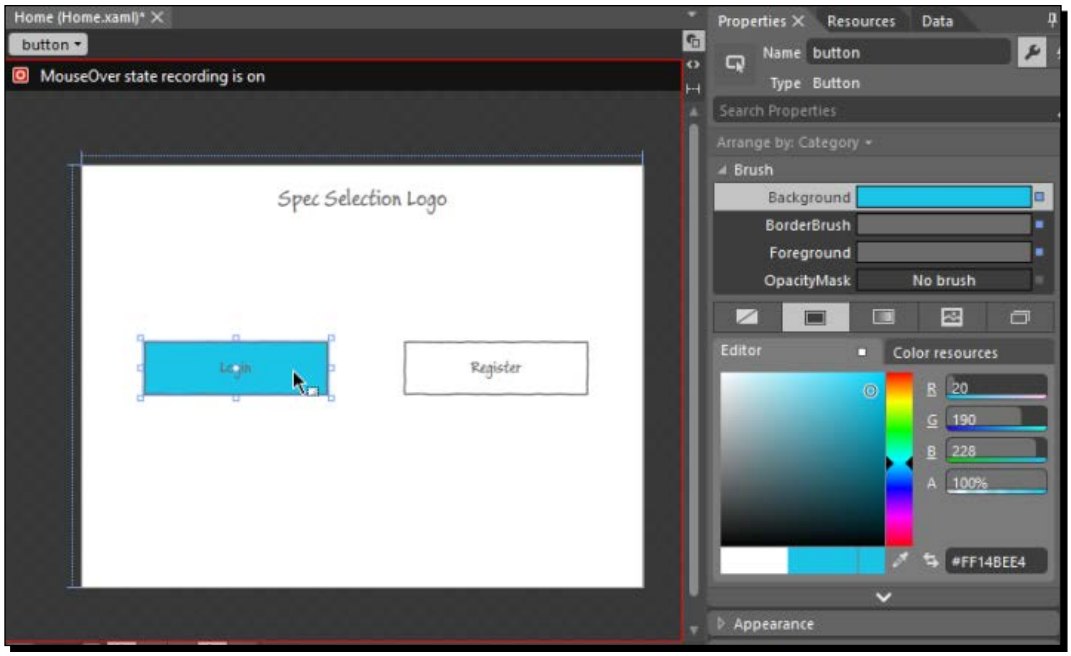
To create animations using states, perform the following steps:

1. Open `Home.xaml` from the **Project** panel. Then, go to the **States** panel and click on **Add state group**.

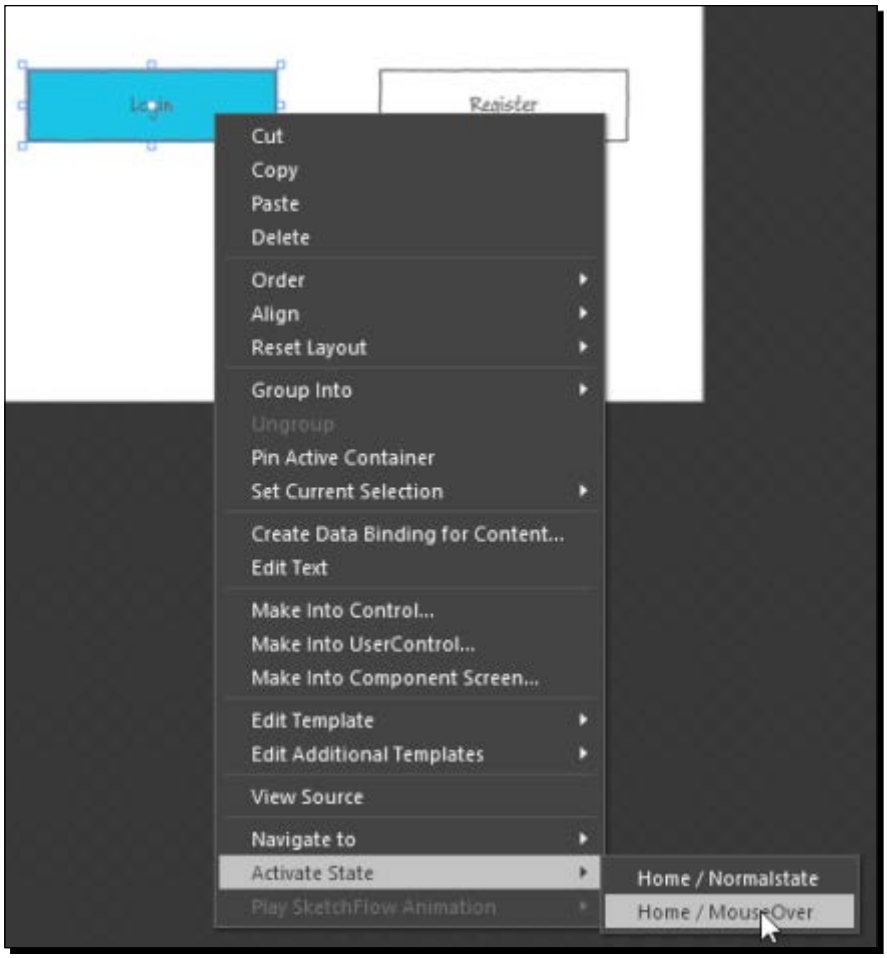
2. Then click on the **Add state** button twice to add two states to **VisualStateGroup**. Rename the visual states to `NormalState` and `MouseOver`. As you might have guessed, `NormalState` is the default or fallback state, and the `MouseOver` state is activated when the mouse is over the button. This is shown in the following screenshot:



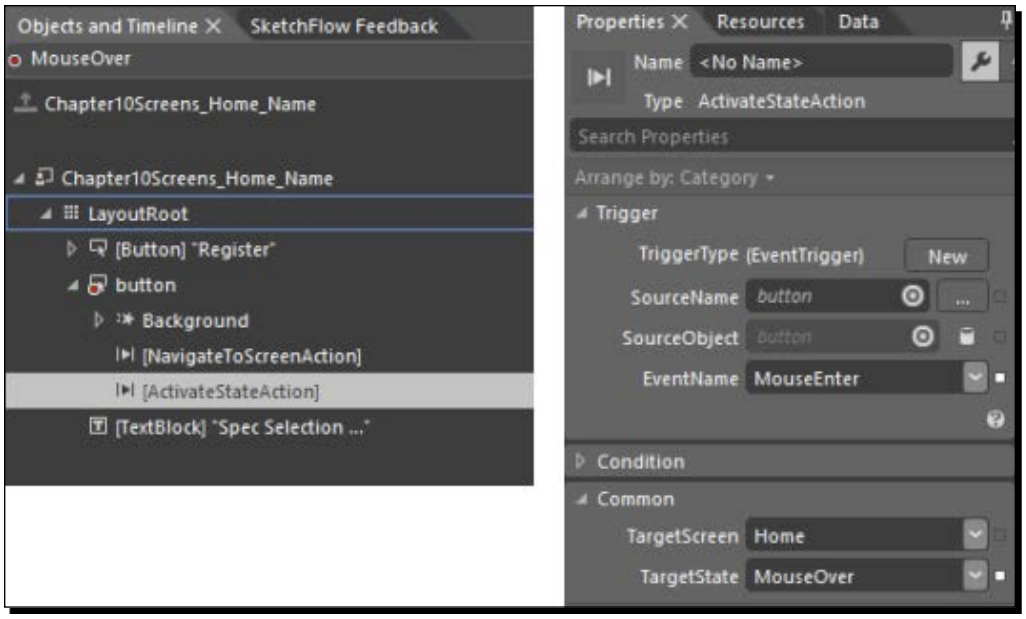
3. We will leave **NormalState** as it is. Select **MouseOver**, and then select the **Login** button. Change **Background** of the **Login** button to blue. This is depicted in the following screenshot:



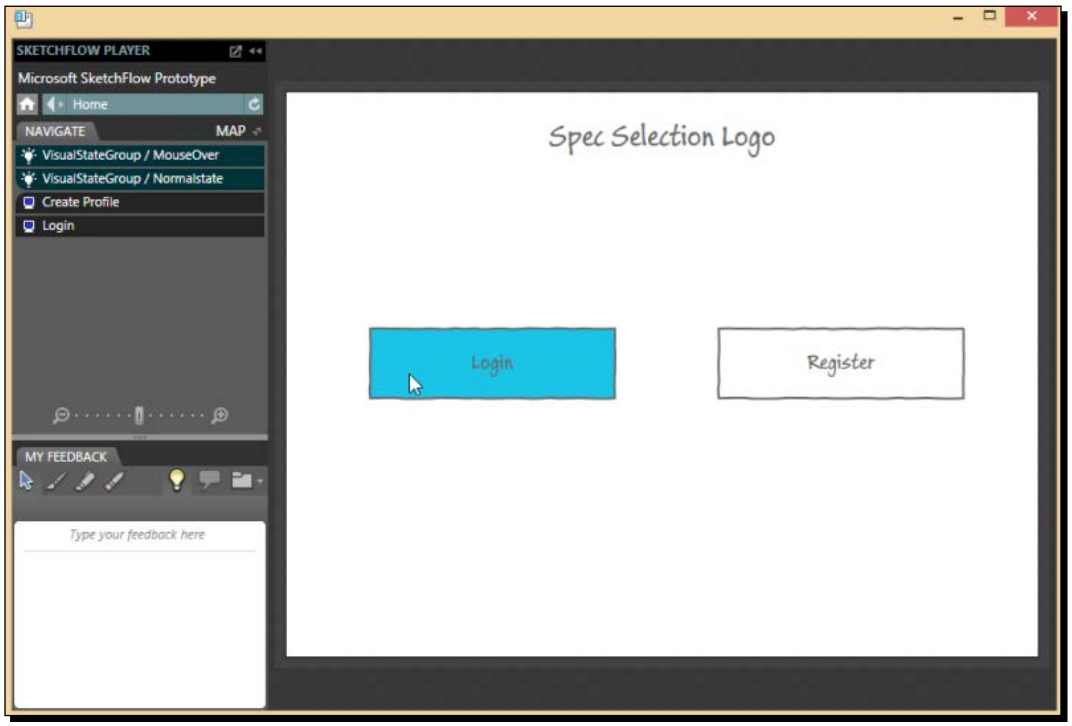
4. Right-click on **Activate State** | **Home / MouseOver**. This will add an **ActivateStateAction** behavior in prototyping. The following screenshot encapsulates this discussion:



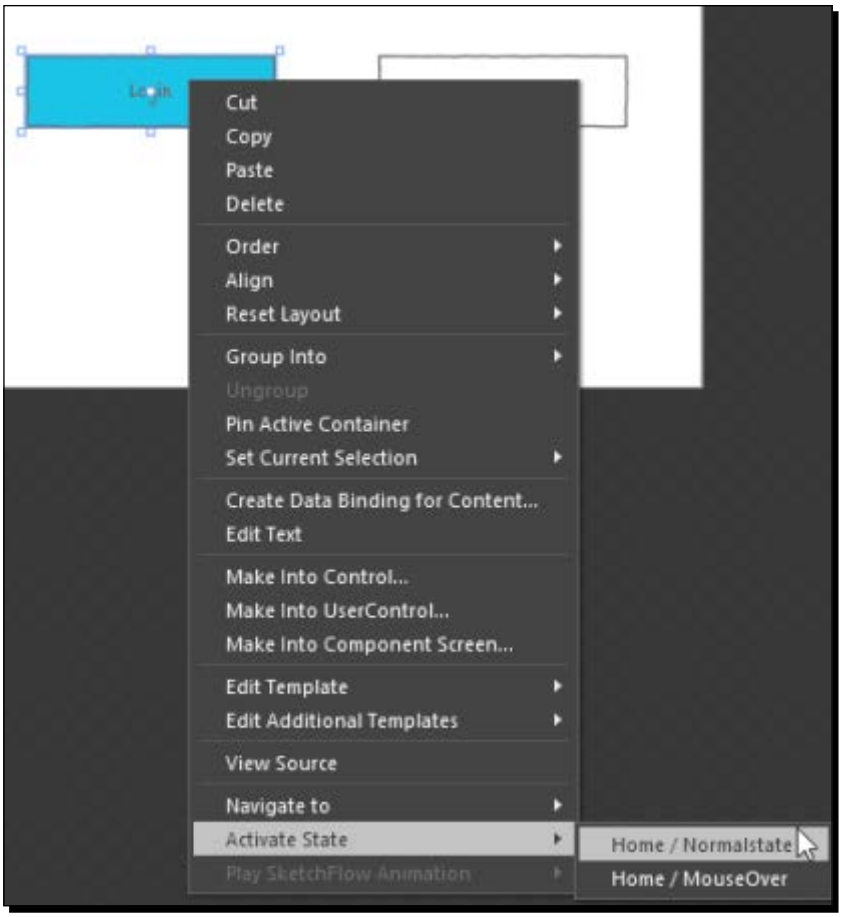
5. This behavior will allow us to configure when this state will be activated. We have specified **EventName** as **MouseOver**. Hence, whenever the mouse enters the button area, the **MouseOver** state will be activated. The following screenshot shows this:



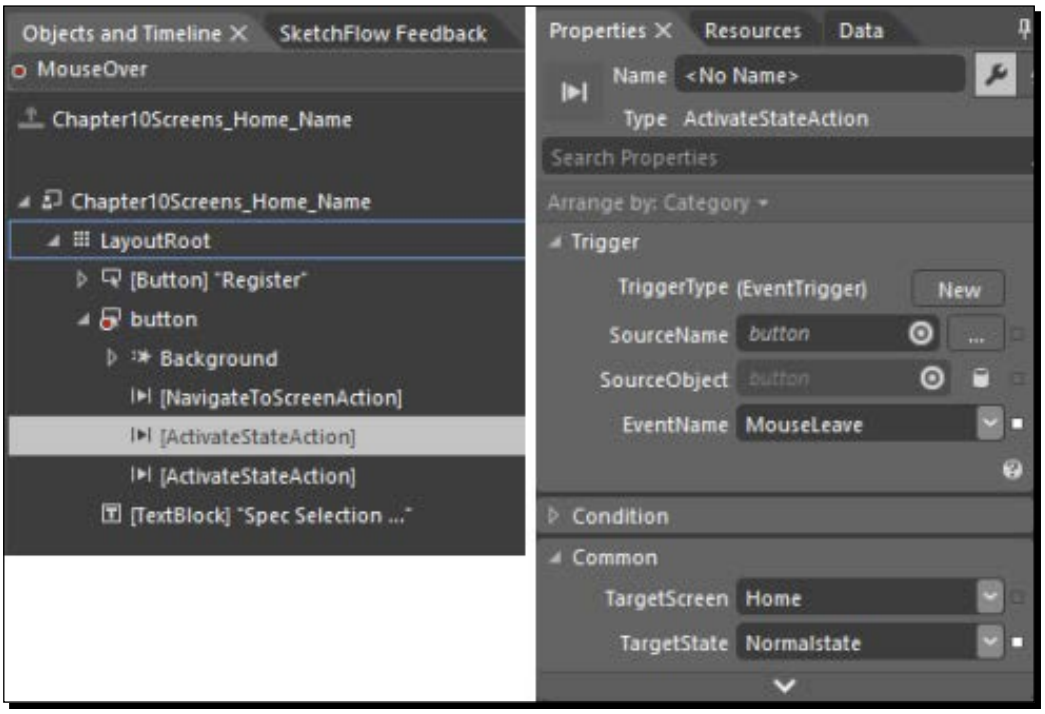
- Now, when you run the application, you will see that when you move the mouse over to the **Login** button, the **MouseOver** state is activated and that when you move the mouse away from the **Login** button, the state is not deactivated. This is shown in the following screenshot:



6. To rectify that behavior, we need to activate Normalstate when the mouse leaves the **Login** button. To do that, right-click on the **Login** button and select **Activate State** | **Home / Normalstate**, as shown in the following screenshot:



7. Set **EventName** to **MouseLeave** and **TargetState** to **NormalState** as shown in the following screenshot:



8. Now, we run the application to take the mouse over to Login, due to which the **MouseOver** state is activated, and when we move the mouse out of the **Login** button area, the button goes back to **NormalState**.

What just happened?

We have the animation with states in SketchFlow.

Data in SketchFlow

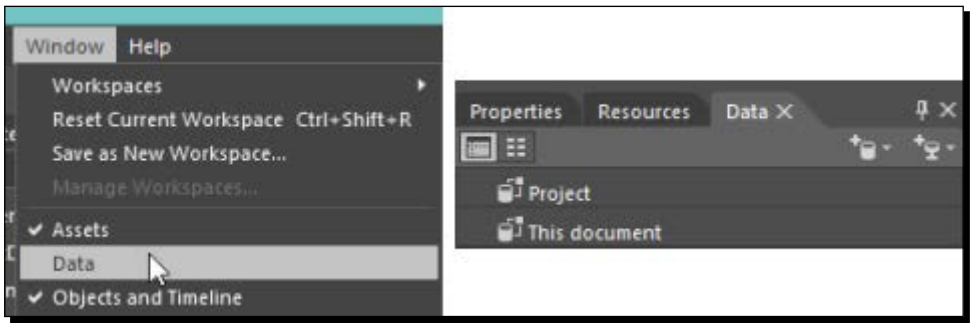
In this section, we will see how to use data with SketchFlow. We could make our prototypes more effective by adding real or simulated data. For example, instead of showing the user a blank form, we could populate it with data and make it more interactive.

Time for action – adding data to SketchFlow

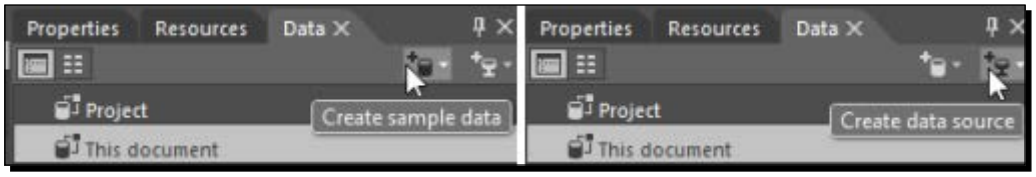
To add data to SketchFlow, perform the following steps:

1. Double-click on the **Frame Selection** screen in SketchFlow map:

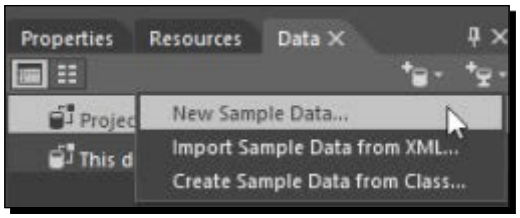
2. Move to the **Data** panel, which is next to the **Resources** panel by default. If it is not already visible, then we could make it visible by navigating to **Window** | **Data**. The following screenshot shows this:



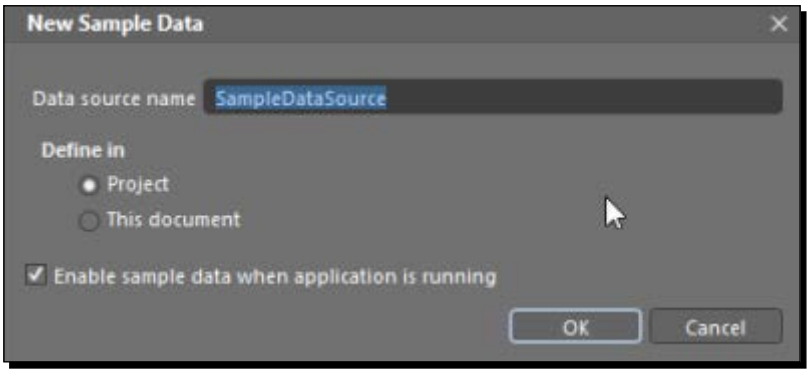
3. In the top-right corner, we see two options; one of them allows us to do **Create sample data** for the application or document, and the other one, **Create data source**, allows us to connect live data sources. This is shown in the following screenshot:



4. Let's add sample data in the application. Project is selected for the data scope. When you click on **Create sample data**, you will get three options; select **New Sample Data** from it, as shown here:

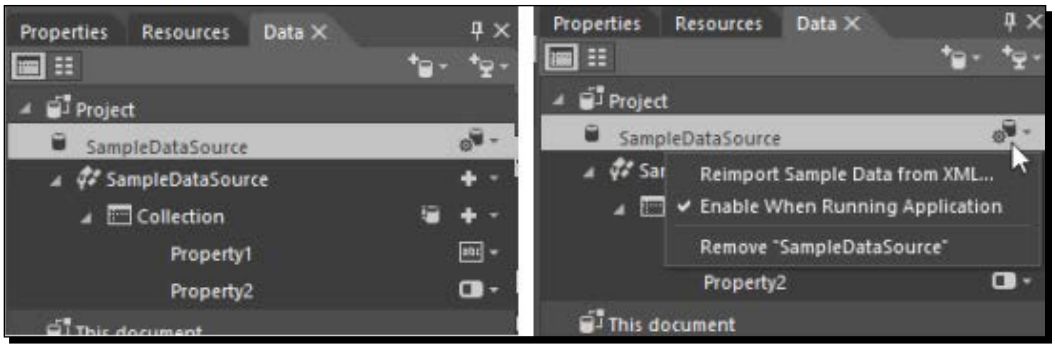


5. We will see a screen asking the name of the data source. You can keep the default name and keep the **Enable sample data when application is running** checkbox checked so that we can access sample data even while running the application. This is shown in the following screenshot:

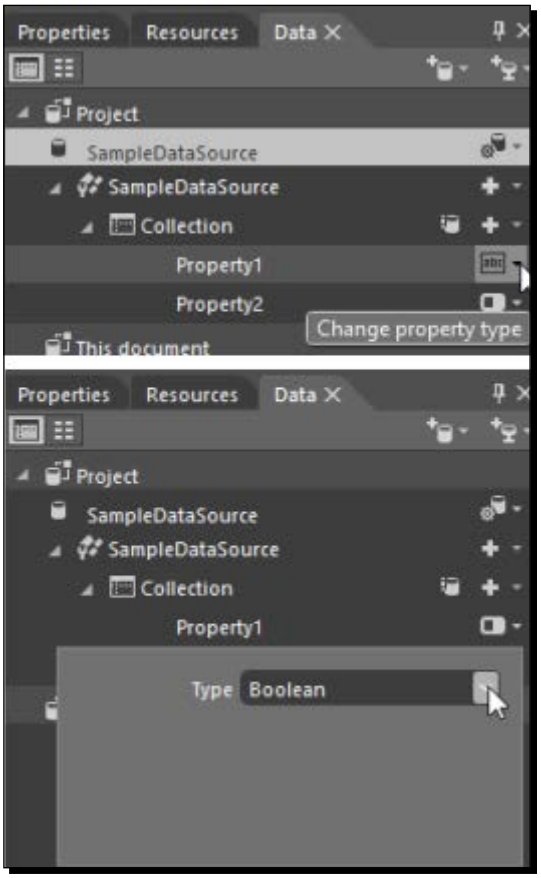


6. Once we click on **OK**, we will see that **SampleDataSource** is added with **Collection** and two properties. We could modify or remove these properties and also add new ones.

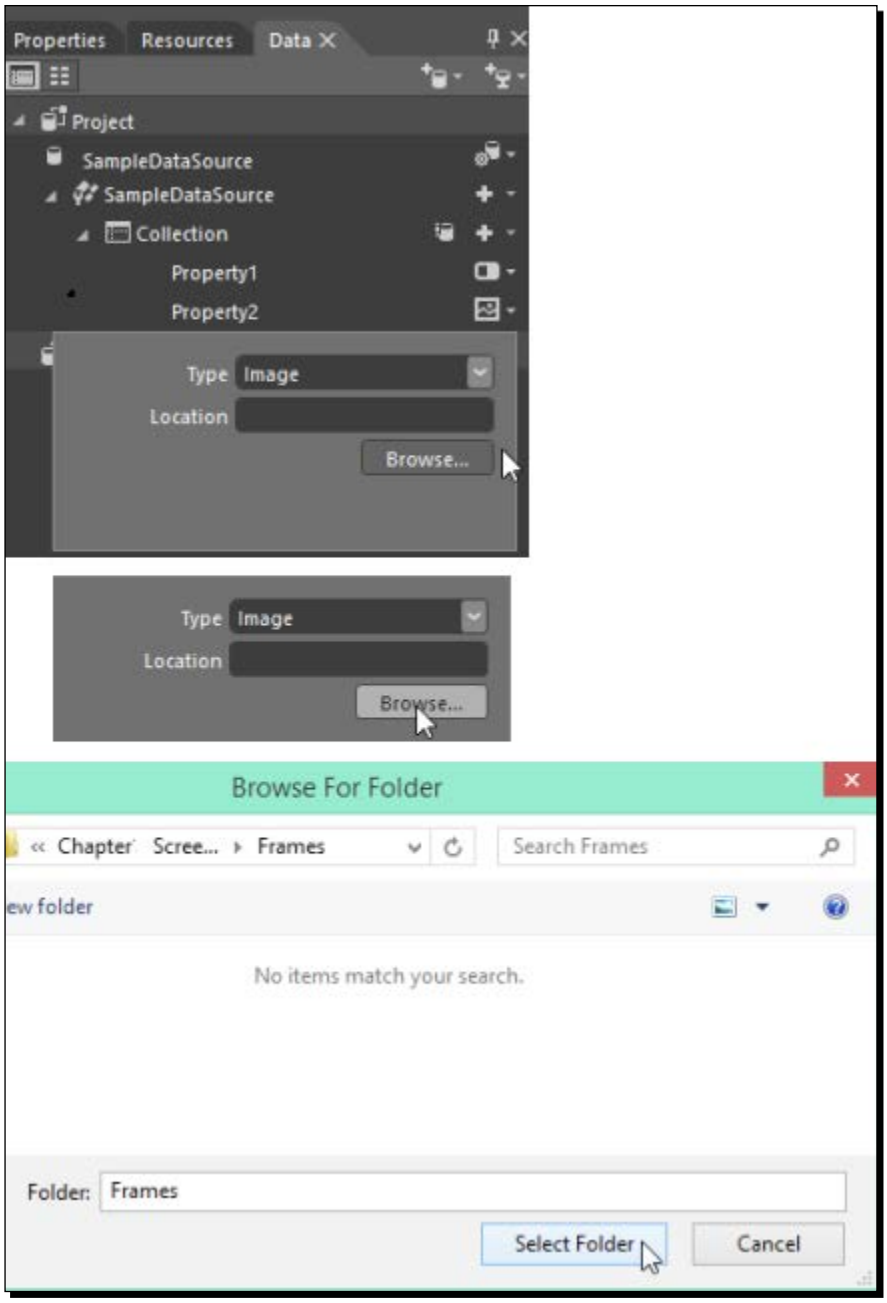
7. Next to **SampleDataSource**, we have a settings button, using which we could use to remove **SampleDataSource**. We have the option to enable or disable this data source when the application is running and also import sample data into this data source from XML. All this is depicted in the following screenshot:



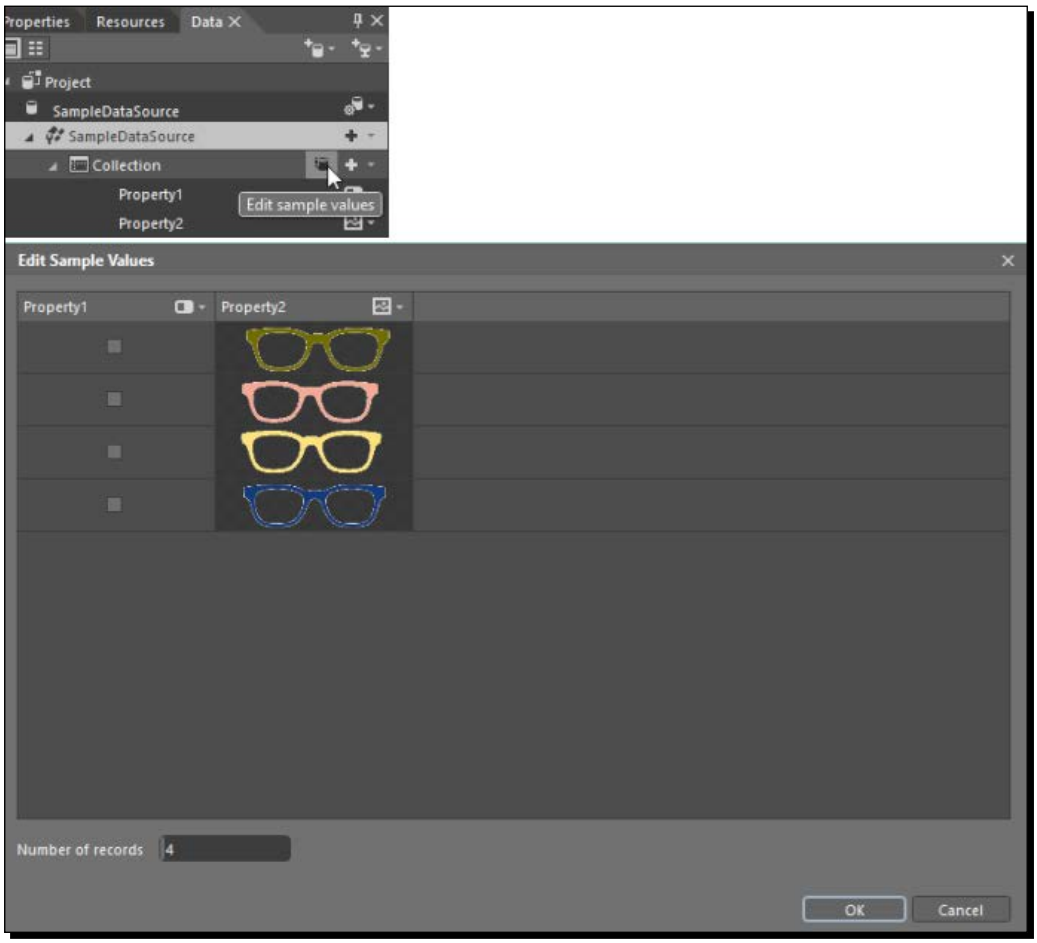
8. We will modify these properties according to our requirements. Click on the dropdown next to **Property1** and change the type of property to **Boolean**, as shown in the following screenshot:



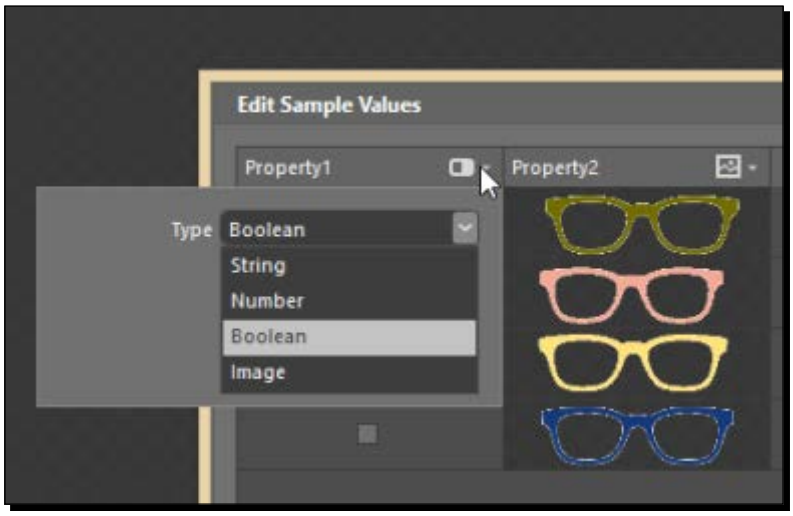
9. Change the type of **Property2** to **Image**, and then browse to a folder with images of frames (create a folder and put a few images in it before browsing). The following screenshot illustrates this:



10. Click on **Edit Sample Values**, and we will get the view to edit the sample values. We have changed **Number of records** to `4` as we have four images of frames. We could even modify the property values of the individual entries. This is shown in the following screenshot:



11. We could change the property values for each property by selecting a data type and the format of each property, as shown in the following screenshot:



What just happened?

We just created sample data that could be used in our application.

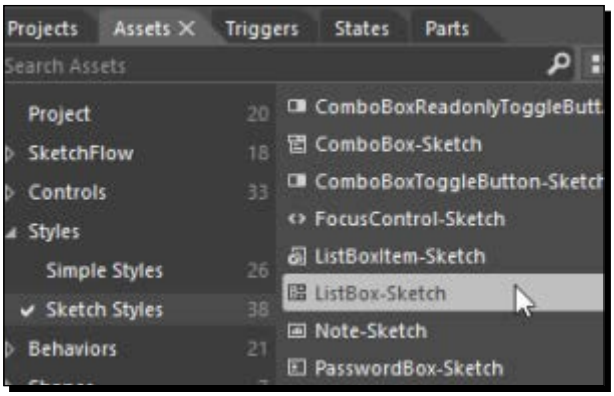
Binding data to control

Now that we have created the data, it's time to use them with our control.

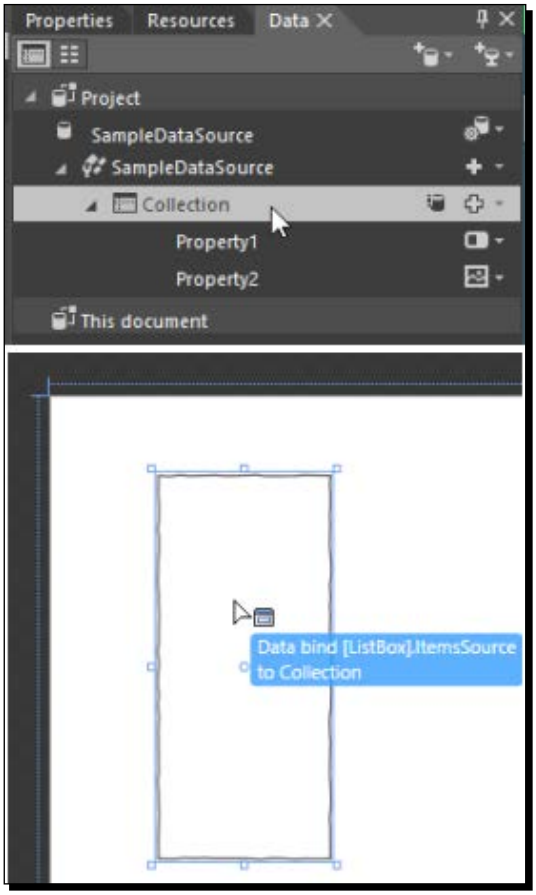
Time for action – adding data to control

To use the data with our control, perform the following steps:

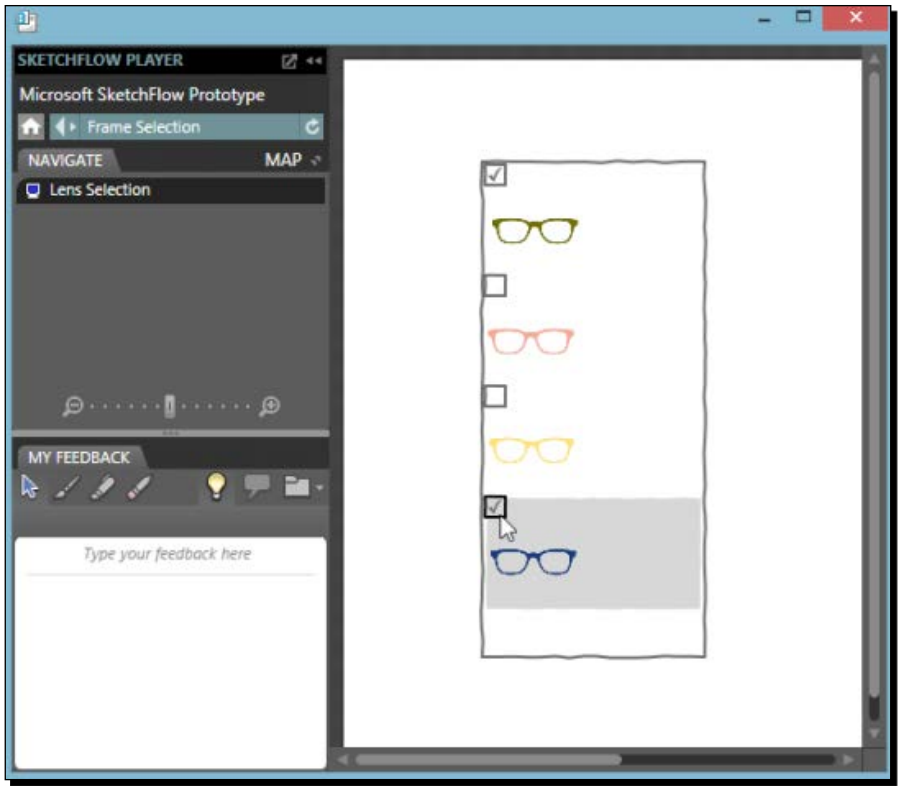
1. Go to the **Assets** panel, and under **Styles** | **Sketch Styles**, select **ListBox-Sketch** and drag it onto the art board. This is shown in the following screenshot:



2. Now, for the **Data** panel, select **Collection**, and drag it onto ListBox. This will bind the data collection with ListView. This is shown in the following screenshot:



3. Now, when we run the application and navigate to the **Frames Selection** screen, we can see ListBox populated with the data. We can select or deselect options for the frame. This is depicted in the following screenshot:



What just happened?

We added data to ListBox and interacted with it at runtime.

Pop quiz

Q1. Is it possible to add animations in SketchFlow?

1. Yes.
2. No.

Q2. Is it possible to use DataBinding in SketchFlow?

1. Yes.
2. No.

Q3. Can we run a SketchFlow application?

3. Yes. It runs within a SketchFlow Player .
4. No, it's not possible.

Summary

In this chapter, we created a SketchFlow application, had it reviewed, and incorporated the feedback. We added animations, states, and data to the application as well.