

Working with View States and Animation and Transitions

Import the Project

1. In Flex Builder, select **File > Import > Flex Project**.
2. In the dialog window, select **Archive File** and browse to where **Ex17_Starter.zip** is located in your local file system.
3. Click **Finish**.

Note: Alternatively you can start with the code you completed from exercise 16.

Display a login page by default

4. In the **Flex Navigator** view of Flex Builder, right click on the **components** folder and select **New > MXML Component**.
 - Filename : Reservation
 - Based on : Canvas
 - Width : 1500
 - Height : remove default value
5. Click **Finish**.
6. Open **ReservationSystem.mxml** and cut all the code after the `Style` tag and before the closing `Application` tag.
7. Paste the code into the **Reservation.mxml** file between the `Canvas` tags.
8. To the opening `Canvas` container tag, add the `creationComplete` property with a value of `init()` and a new namespace named `comp` that points to all files in the `components` folder.

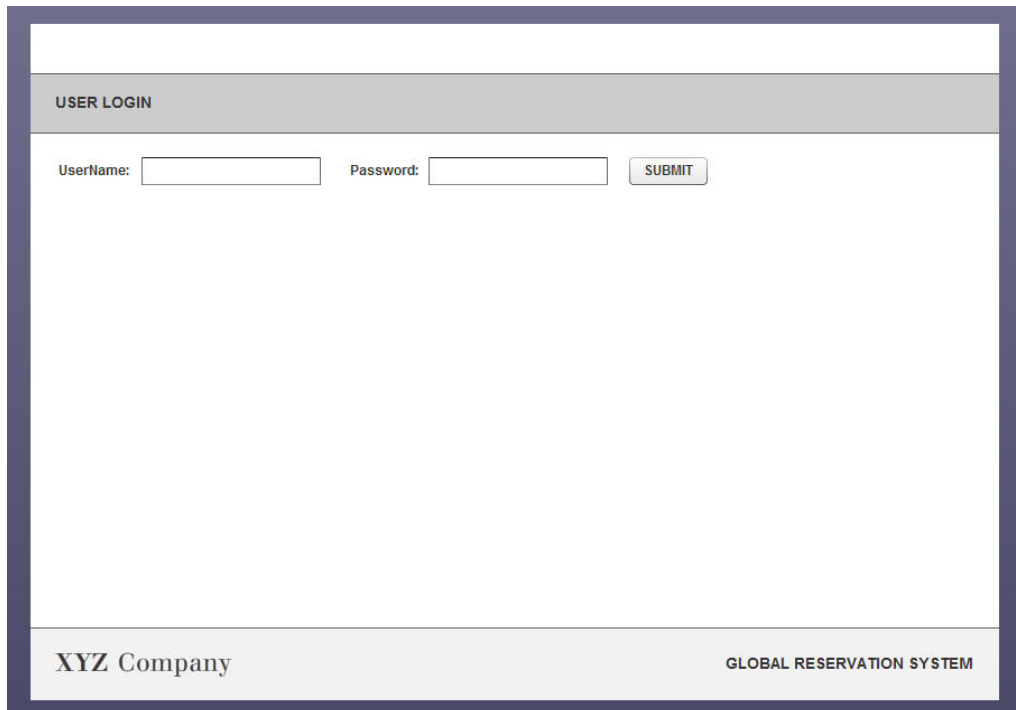
```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
width="1500"
creationComplete="init()"
xmlns:comp="components.*">
```

9. Save the file.
10. Return to **ReservationSystem.mxml**.
11. Add a `horizontalScrollPolicy` property with a value of `off` to the opening `Application` tag.

12. After the `Style` tag that references the CSS file, add an instance of the `Login` component from the `comp` namespace.
13. Add an `id` property with a value of `login`, a `width` property with a value of `864`, a `height` property with a value of `604`, a `horizontalCenter` with a value of `0` (zero), a `verticalCenter` with a value of `0` (zero) and a `y` property with a value of `20`.

```
<comp:Login id="login"
            width="864" height="604"
            horizontalCenter="0" verticalCenter="0"
            y="20"/>
```

14. Save the file and run the application.
You should see the login page however you will not be able to log on yet.

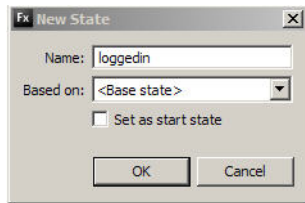


The screenshot shows a web application window with a dark blue border. At the top, there is a grey header bar with the text "USER LOGIN". Below the header, the main content area is white and contains a login form. The form has two labels: "UserName:" and "Password:", each followed by a rectangular text input field. To the right of the password field is a button labeled "SUBMIT". At the bottom of the window, there is a grey footer bar. On the left side of the footer, it says "XYZ Company", and on the right side, it says "GLOBAL RESERVATION SYSTEM".

Create a login view state

15. Return to the **ReservationSystem.mxml** file.
16. Add the `creationComplete` property with a value of `init()` to the beginning `Application` tag.
17. Switch to the **Design** view.

18. In the **States** panel, right click on the **<Base state>** and select **New State**.
19. Enter **loggedin** and click **OK**.



20. Using the **Outline** view, select the **Login** component.
Note: You may need to resize your **Design area** to see the selected component.
21. Click the highlighted component on the stage and delete it.
You are deleting everything on the stage as a group.
22. Switch to **Source** view. You will see the `states` code block written for you below the opening `Application` tag.

```
<mx:states>
  <mx:State name="loggedin">
    <mx:RemoveChild target="{login}"/>
  </mx:State>
</mx:states>
```

23. After the `RemoveChild` tag between the `State` tags, add an `AddChild` tag set.
24. Between the `AddChild` tags, add the `Reservation` components from the `comp` namespace.
25. Add a `horizontalCenter` property with a value of 0 (zero) and a `y` property with a value of 20. You are adding a different display on the stage that replaces the login display.

```
<mx:states>
  <mx:State name="loggedin">
    <mx:RemoveChild target="{login}"/>
    <mx:AddChild>
      <comp:Reservation horizontalCenter="0"
        verticalCenter="0" />
    </mx:AddChild>
  </mx:State>
</mx:states>
```

26. After the beginning `Application` tag, create a `Script` block.

27. Create a private function named `init()` that takes no parameters and returns `void`.
28. Within the function add an event listener for the `login` component that listens for the `login` event and executes the `loginHandler` function.
Note: Remember that you do not reference the function with parentheses in this case.

```
login.addEventListener("login", loginHandler);
```

29. Create a private function named `loginHandler()` that takes an event parameter datatyped as `Event` and returns `void`.
30. Within the function assign the literal value `loggedin` to `this.currentState`.

You code should appear as follows:

```
private function init():void {  
    login.addEventListener("login", loginHandler);  
}  
  
private function loginHandler(event:Event):void {  
    this.currentState = "loggedin";  
}
```

31. Save the file and run the application.
32. Click the submit button.
You should see the state change from the login to the reservation view.
Note: If you receive an error message stating that there is a security setting violation or that the application can't access a resource or asset, then you will need to add a compiler setting that will allow you to access resources from the local file system. Use the following steps to add the compiler setting.
 - a. Right click on the **ReservationSystem** project and select **Properties** from the menu.
 - b. Select the **Flex Compiler** menu.
 - c. Add a space and then add **-use-network=false** to the compiler arguments.
 - d. Click OK.
33. Return to the **Reservation.mxml** file.

34. Add a `horizontalScrollPolicy` property with a value of `off` to the beginning `Canvas` tag. Since the width is 1500 pixels, scroll bars would normally be displayed but in this application you will control the horizontal scrolling by handling a click event.

```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
    creationComplete="init()"
    xmlns:comp="components.*"
    width="1500"
    horizontalScrollPolicy="off">
```

35. Using the **Outline** view, locate the second **Canvas** container.
36. After the `Canvas` tag block, add the `CustomerService` component from the `comp` namespace.
37. Add an `id` property with a value of `cs`, a `width` property with a value of 864 and a `verticalCenter` property with a value of 16.

```
<comp:CustomerService id="cs"
    width="864"
    verticalCenter="16" />
```

38. Open **CustomerService.mxml**.
39. Using the **Outline** view, locate the **Image** component. Notice that the `Image` component has a `click` event that calls an `openCloseForm()` handler that dispatches an `openClose` event.
40. Return to **Reservation.mxml**.
41. Locate the `init()` method in the `Script` block.
42. After the existing code, call the `resizeHandler()` function. You will create this function later.
43. Next, within the `init()` method, add an event listener to the `cs` component that listens for the `openClose` event and executes the `openCloseHandler`. You need to know when the user clicks the image so that you can display the form.

```
cs.addEventListener("openClose", openCloseHandler);
```

44. Create another event listener for the `stage` property that handles a `RESIZE` event and executes the `resizeHandler` function. This listener indicates when the stage changes size. If the size changes you need to relocate the `CustomerService` tab.

```
stage.addEventListener(Event.RESIZE, resizeHandler);
```

45. Below the `init()` function, create private function named `resizeHandler` that takes one parameter named `event` data typed to the `Event` class and set to a value of `null`. The function returns `void`.
Note: Setting a default value of `null` allows you to use the function even when the `Event` object does not exist.

```
private function resizeHandler(event:Event=null):void {  
  
}
```

46. Within the `resizeHandler()` function, after the existing code, assign the `stage.stageWidth` to the `stagewidth` variable that has already been created. This tells you how wide the browser window is.

```
stagewidth = stage.stageWidth;
```

47. Next assign the `stagewidth` variable minus 25 to the `csX` variable that is already created. You subtract 25 pixels to allow the tab to display on the stage while the rest of the component remains off-stage.

```
csX = stagewidth - 25;
```

48. Locate the `CustomerService` component at the end of the file and add an `x` property bound to the `csX` variable. This value determines where the `CustomerService` component is placed in the browser window.

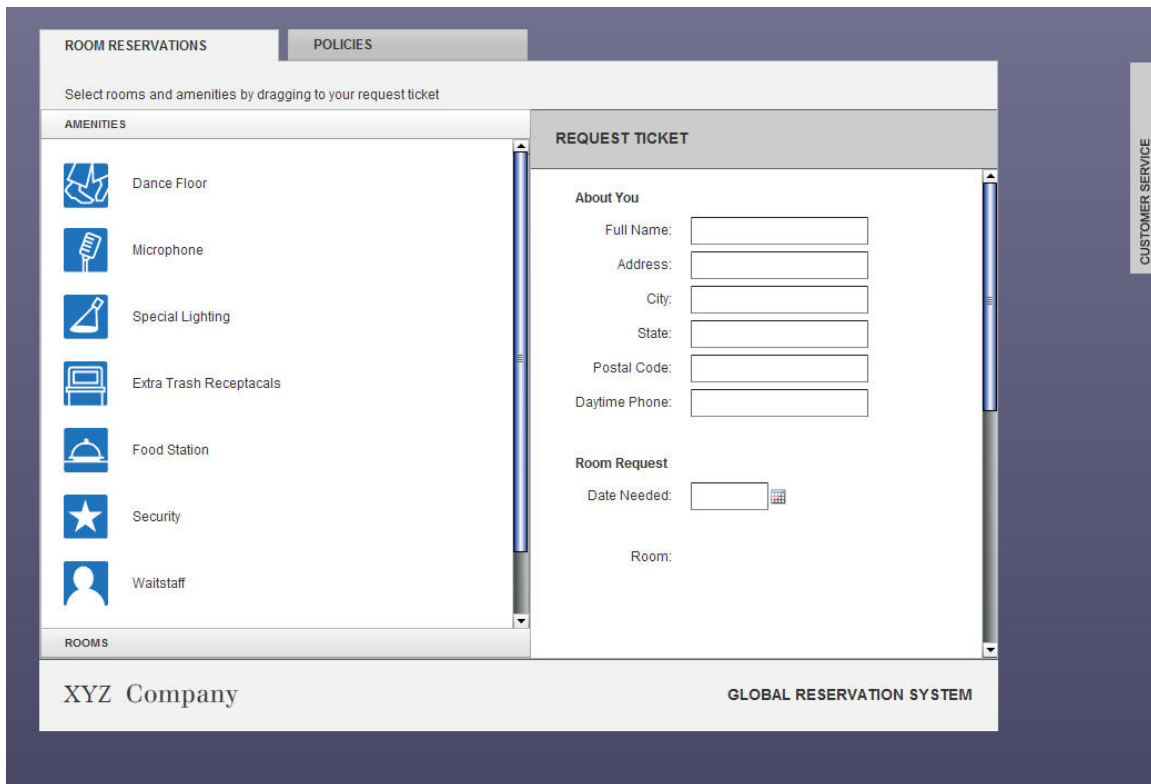
```
<comp:CustomerService id="cs"  
    width="864"  
    verticalCenter="16"  
    x="{csX}" />
```

49. Locate the beginning `Canvas` tag and change the `width` property from 1500 to a binding to the `stagewidth` variable.

```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"  
    width="{stagewidth}"  
    creationComplete="init()"  
    xmlns:comp="components.*"  
    horizontalScrollPolicy="off">
```

50. Save the file and run the application.
You should see that after you log on, the customer service page tab appears on the right side of the browser window. You will create a view state to see the customer service page next.

Note: To see the customer service page tab, please use full screen mode on the browser window. Sometimes, the vertical scroll bar can hide the tab from view.



Create the customer service view state

51. Return to the **Reservation.mxml** file.
52. Using the **Outline** view, locate the second **Canvas** container.
53. Add an `id` property with a value of `canvas1`.

```
<mx:Canvas id="canvas1"
           horizontalCenter="0" verticalCenter="0">
```
54. Using the **Outline** view, locate the first **Canvas** container.
55. After the beginning of the first **Canvas** container and before the `Script` tag block, create a `states` tag set.

56. Between the `states` tags, add a `State` tag set with a `name` property set to a value of `csOpen`.

```
<mx:states>
    <mx:State name="csOpen">
    </mx:State>
</mx:states>
```

57. Between the `State` tags, create a `SetStyle` tag with a `target` property bound to `canvas1`, a `name` property with a value of `horizontalCenter` and a `value` property with a value of `-409`.

```
<mx:SetStyle target="{canvas1}"
    name="horizontalCenter" value="-409"/>
```

58. After the `SetStyle` tag, add a `SetProperty` tag with a `target` property bound to `cs`, a `name` property with a value of `x` and a `value` property with a value of `818`.

```
<mx:SetProperty target="{cs}"
    name="x" value="618"/>
```

59. Create a `SetStyle` tag with a `target` property bound to `cs`, a `name` property with a value of `verticalCenter` and a `value` property with a value of `13`.

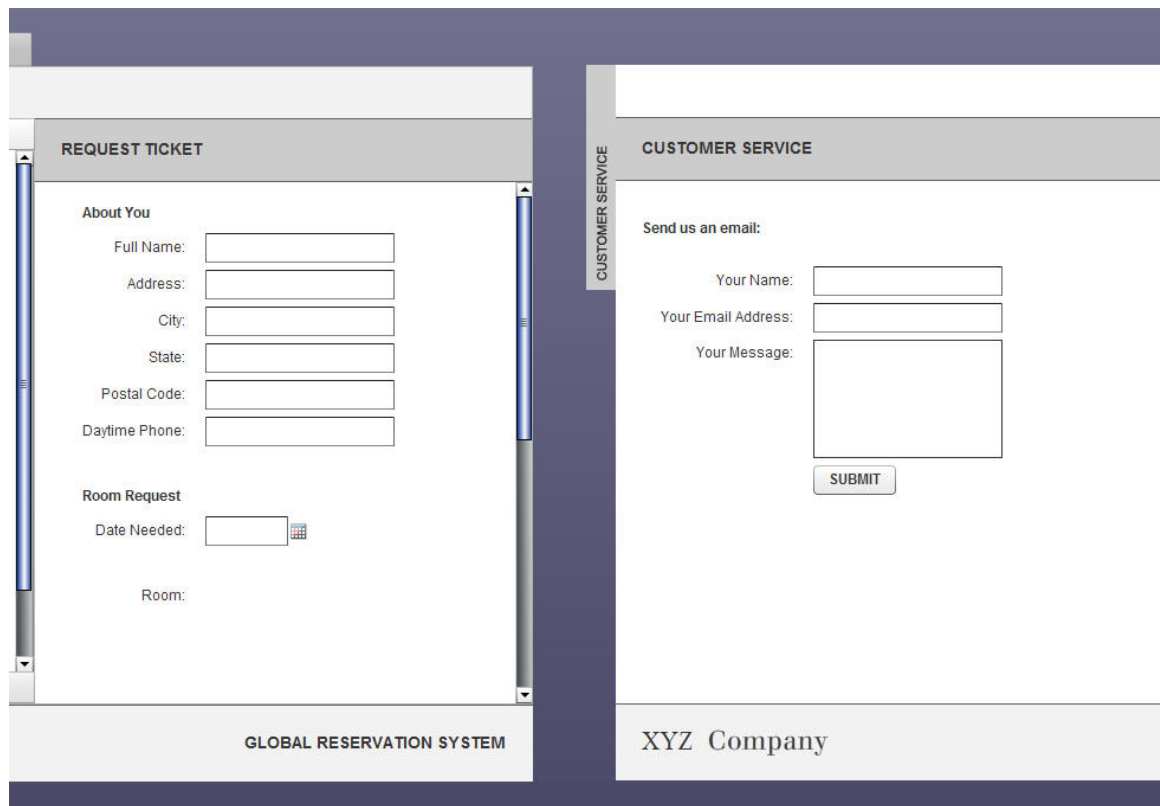
```
<mx:SetStyle target="{cs}"
    name="verticalCenter" value="13"/>
```

Your code should appear as follows:

```
<mx:states>
    <mx:State name="csOpen">
        <mx:SetStyle target="{canvas1}"
            name="horizontalCenter" value="-409"/>
        <mx:SetProperty target="{cs}"
            name="x" value="618"/>
        <mx:SetStyle target="{cs}"
            name="verticalCenter" value="13"/>
    </mx:State>
</mx:states>
```

```
</mx:State>
</mx:states>
```

60. Save the file and run the application.
You should see the customer service tab after you log in.
61. Click on the tab.
You should see the application shift to the left to reveal the Customer Service form.



Display the room list images

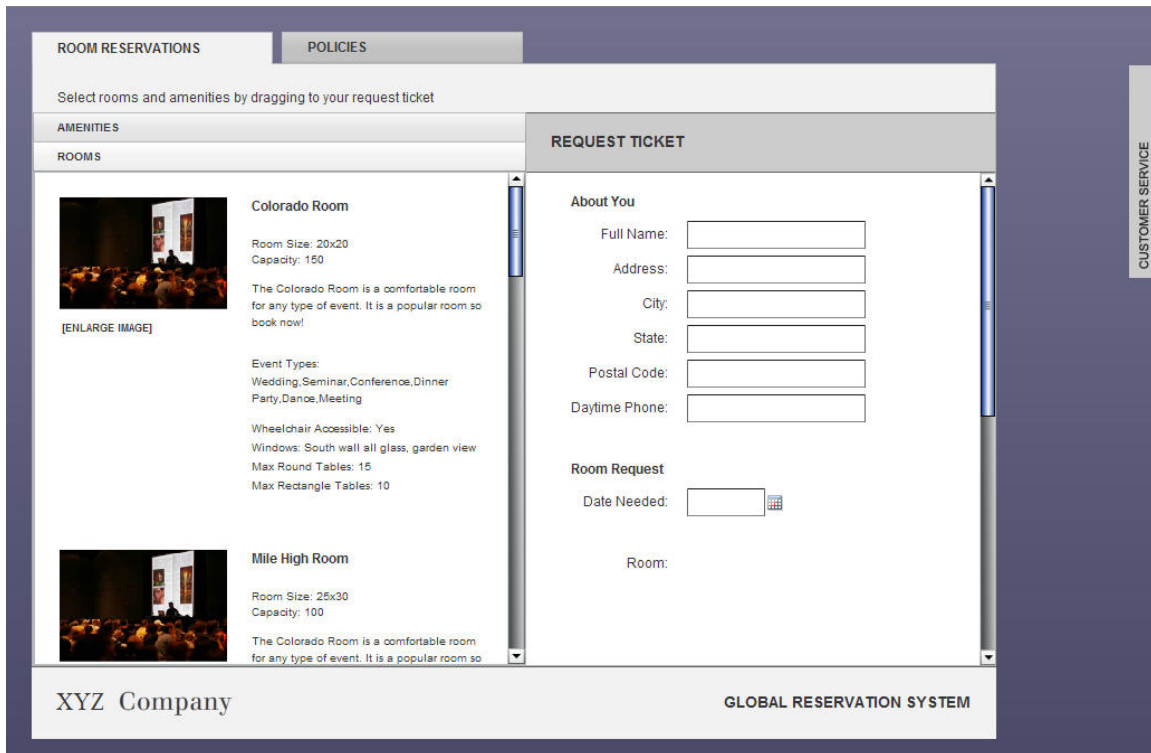
62. Return to the **Reservation.mxml** file.
63. Using the **Outline** view, locate the **TabNavigator** instance.
64. Locate the **RoomReservations** instance nested inside the **TabNavigator** component.
65. Add a **roomList** property that is bound to the **roomList** variable.

```
<comp:RoomReservations label="ROOM RESERVATIONS"
    roomList="{roomList}" />
```

66. Save the file and run the application.
67. Click the submit button in the login form.
68. Click the Rooms pleat of the **Accordion** control.
You should see the rooms listed in the **Rooms** panel of the accordion.
69. Open the **RoomRenderer.mxml** file in the **components** folder.
70. Between the two closing `Canvas` tags at the end of the document, add a `Thumbnail` component instance from the `comp` namespace.
71. Add a property named `id` with a value of `tb`, a `y` property with a value of `20` and an `x` property with a value of `20`.

```
<comp:Thumbnail id="tb"
                y="20" x="20" />
```

72. Save the file.
73. Open the **RoomReservations.mxml**.
74. Using the **Outline** view, locate the **Accordion** instance.
75. Add a `selectedIndex` property to the opening `Accordion` tag with a value of `1`.
76. Save the file and run the application.
77. Click the submit button on the login page and open the Rooms pleat in the **Accordion** control.
You should see the Rooms pleat open and images displayed.



Enlarge the image using View States

78. Open the **Thumbnail.mxml** component from the **components** folder. Notice that five bindable variables have already been created for you.
79. In **beginning** Canvas tag change the values to bind the width to `thumbnailWidth` and bind the height to `containerHeight`.

```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
    width="{thumbnailWidth}"
    height="{containerHeight}">
```

80. Using the **Outline** view, locate the Image component.
81. Change the width value to bind to `smallImageWidth` and bind the height to `smallImageHeight`. This is the default display size.

```
<mx:Image id="thumbnailImage"
    source="assets\conference.jpg"
    width="{smallImageWidth}"
    height="{smallImageHeight}"/>
```

82. Switch to the **Design** mode.

Note: The image in Design mode is larger than the image that you saw in the browser. Therefore, you cannot easily see the Label control.

83. In the States panel, right-click on the base state and select **New State**.
84. Name the state **expand** and click **OK**.
85. Use Outline view to locate the **msg** Label control and delete it so that it doesn't appear when the user selects the expand state.
86. Return to **Source** mode.
87. Locate the `states` code at the top of the file that has been generated for you.

```
<mx:states>
  <mx:State name="expand">
    <mx:RemoveChild target="{msg}"/>
  </mx:State>
</mx:states>
```

88. After the `RemoveChild` tag, add a `SetProperty` tag with a target property set to a value bound to `thumbnailImage`.
89. Add a name property with a value of `width` and a value property bound to `thumbnailWidth`. This is the width the image will expand to.

```
<mx:SetProperty target="{thumbnailImage}"
  name="width" value="{thumbnailWidth}" />
```

90. Add another `SetProperty` tag with a target property set to a value bound to `thumbnailImage`.
91. Add a name property with a value of `height` and a value property bound to `thumbnailHeight`. This is the height the image will expand to.

```
<mx:SetProperty target="{thumbnailImage}"
  name="height" value="{thumbnailHeight}" />
```

92. After the `SetProperty` tag, add an `AddChild` tag block.
93. Between the `AddChild` tags, create a `Label` tag with an `id` property set to a value of `msg2`.
94. Also add the following properties and values:

- `y : 238`
- `textAlign : right`
- `text : [SHRINK IMAGE]`
- `fontWeight : bold`
- `fontSize : 9`
- `right : 0`

Your code should appear as follows:

```
<mx:states>
  <mx:State name="expand">
    <mx:RemoveChild target="{msg}"/>
    <mx:SetProperty target="{thumbnailImage}"
      name="width" value="{thumbnailWidth}"/>
    <mx:SetProperty target="{thumbnailImage}"
      name="height"
      value="{thumbnailHeight}"/>
    <mx:AddChild>
      <mx:Label id="msg2"
        y="238"
        textAlign="right"
        text="[SHRINK IMAGE]"
        fontWeight="bold" fontSize="9"
        right="0"/>
    </mx:AddChild>
  </mx:State>
</mx:states>
```

95. Using the **Outline** view, locate the **msg** Label.
96. Add a click event with a value of `current='expand'`.

```
<mx:Label id="msg" y="109" textAlign="right"
  text="[ENLARGE IMAGE]"
  fontWeight="bold" fontSize="9" left="0"
  click="currentState='expand'" />
```

97. Save the file and run the application.
98. Click the submit button on the login page.
99. Click the label **Enlarge Image** under one of the images in the Rooms pleat of the Accordion container.
You should see it enlarge over the text.
100. Return to the **Thumbnail.mxml** file.
101. Using the **Outline** view, locate the **msg** Label control and add the `expand()` method to the `click` event. The `expand()` method is already created for you.

```
<mx:Label id="msg" y="109" textAlign="right"
  text="[ENLARGE IMAGE]"
  fontWeight="bold" fontSize="9"
```

```
click="currentState='expand'; expand()" left="0"/>
```

102. Locate the **msg2** Label control in the states block.
103. Create a click event with a value of `currentState=''` and also call the `shrink()` method. The shrink method is already created for you.

```
<mx:Label id="msg2" y="238"  
    textAlign="right"  
    text="[SHINK IMAGE]"  
    fontWeight="bold" fontSize="9"  
    right="0"  
    click="currentState='';shrink()"/>
```

104. Save the file.
105. Return to **RoomRenderer.mxml** and locate the Thumbnail component instance at the bottom of the page.
106. Add an expand event with a value of `expandHandler(event)` and a shrink event with a value of `shrinkHandler(event)`.

```
<comp:Thumbnail id="tb" y="20" x="20"  
    expand="expandHandler(event)"  
    shrink="shrinkHandler(event)"/>
```

107. Save the file and run the application.
108. Click the submit button.
109. Click the Enlarge image label in the **Rooms** accordion panel.
You should see the image expand and the text disappear.
110. Click the **Shrink image** label to shrink the image.
You should see the image shrink and the text reappear.

Add a transition to display the Customer Service page

111. Open **Reservation.mxml** from the **components** folder.
112. After the states code, create a transitions tag block.
113. Between the transitions tags, create a Transition tag block with a property named `fromState` set to a value of "*" and a `toState` property set to a value of "*".

```
<mx:transitions>
```

```
    <mx:Transition fromState="*" toState="*">
    </mx:Transition>
</mx:transitions>
```

114. Between the Transition tags, create a Parallel tag block.
115. Between the Parallel tags, create a Move effect with a duration property set to a value of 1200 and a target property bound to the cs control.

```
<mx:Move duration="1200" target="{cs}" />
```

116. Create a Move tag with a duration property set to a value of 1200 and a target property set to a value of canvas1.

```
<mx:Move duration="1200" target="{canvas1}" />
```

117. Create an AnimateProperty tag with a target property bound to cs and a property set to a value of x.

```
<mx:AnimateProperty target="{cs}" property="x" />
```

118. Create an AnimateProperty tag with a target property bound to cs and a property set to a value of width.

```
<mx:AnimateProperty target="{cs}" property="width" />
```

119. Your code should appear as follows:

```
<mx:transitions>
  <mx:Transition fromState="*" toState="*">
    <mx:Parallel>
      <mx:Move duration="1200"
        target="{cs}" />
      <mx:Move duration="1200"
        target="{canvas1}"/>
      <mx:AnimateProperty target="{cs}"
        property="x"/>
      <mx:AnimateProperty target="{cs}"
        property="width"/>
    </mx:Parallel>
  </mx:Transition>
</mx:transitions>
```

120. Save the file and run the application.

121. Click the submit button to log into the application.
122. Click the customer service tab.
You should see the page shift smoothly to the left to display the Customer Service form.
123. Click the tab again.
You should see the page returns smoothly.

Add a behavior to the image

124. Open **Thumbnail.mxml** from the **components** folder.
125. Using the **Outline** view, locate the `states` code.
126. After the `states` code, create a `transitions` tag block.
127. Between the `transitions` tags, create a `Transition` tag block with an `id` property set to a value of `expandImage`.
128. Add a `fromState` property with a value of `"*"` and a `toState` with a value of `expand`.

```
<mx:transitions>
    <mx:Transition id="expandImage"
        fromState="*" toState="expand">
    </mx:Transition>
</mx:transitions>
```

129. Between the `Transition` tags, create a `Parallel` tag set with a `target` property bound to the `thumbnailImage` control.
130. Between the `Parallel` tags, create a `Resize` tag with the following properties bound to the listed values.
 - `widthFrom:smallImageWidth`
 - `widthTo:thumbnailWidth`
 - `heightFrom:smallImageHeight`
 - `heightTo:thumbnailHeight`

131. Your code should appear as follows:

```
<mx:transitions>
```

```

<mx:Transition id="expandImage"
fromState="*" toState="expand">
    <mx:Parallel target="{thumbnailImage}">
        <mx:Resize
            widthFrom="{smallImageWidth}"
            widthTo="{thumbnailWidth}"
            heightFrom="{smallImageHeight}"
            heightTo="{thumbnailHeight}"/>
        </mx:Parallel>
    </mx:Transition>
</mx:transitions>

```

132. After the Transition tag, but within the transitions block, create another Transition tag set with an id property set to a value of restoreImage.
133. Add a fromState property with a value of expand and a toState property with a value of "*".
134. Between the Transition tags create a Parallel tag set with a target property bound to thumbnailImage.
135. Between the Parallel tags, create a Resize tag with the following properties bound to the listed values:
 - widthFrom:thumbnailWidth
 - widthTo:smallImageWidth
 - heightFrom:thumbnailHeight
 - heightTo:smallImageHeight

136. Your code should appear as follows:

```

<mx:transitions>
    <mx:Transition id="expandImage"
        fromState="*" toState="expand">
        <mx:Parallel target="{thumbnailImage}">
            <mx:Resize
                widthFrom="{smallImageWidth}"
                widthTo="{thumbnailWidth}"

```

```

        heightFrom="{smallImageHeight}"
        heightTo="{thumbnailHeight}" />
    </mx:Parallel>
</mx:Transition>
<mx:Transition id="restoreImage"
    fromState="expand" toState="*">
    <mx:Parallel target="{thumbnailImage}">
        <mx:Resize
            widthFrom="{thumbnailWidth}"
            widthTo="{smallImageWidth}"
            heightFrom="{thumbnailHeight}"
            heightTo="{smallImageHeight}" />
    </mx:Parallel>
</mx:Transition>
</mx:transitions>

```

137. Save the file and run the application.
138. Click the submit button to log into the application.
139. Click to enlarge an image.
You should see the image smoothly enlarge.
140. Click to shrink the image.
You should see the image smoothly shrink.

Select rooms and amenities by dragging to your request ticket

AMENITIES

ROOMS



[SHINK IMAGE]



Mile High Room

Room Size: 25x30
Capacity: 100

The Colorado Room is a comfortable room for any type of event. It is a popular room so

REQUEST TICKET

About You

Full Name:

Address:

City:

State:

Postal Code:

Daytime Phone:

Room Request

Date Needed: 

Room: