



HCL Volt MX

コントラクトありの コンポーネント

HVMX-BC-500 Volt MX Iris Collaboration

コントラクトを持つコンポーネント – 機能

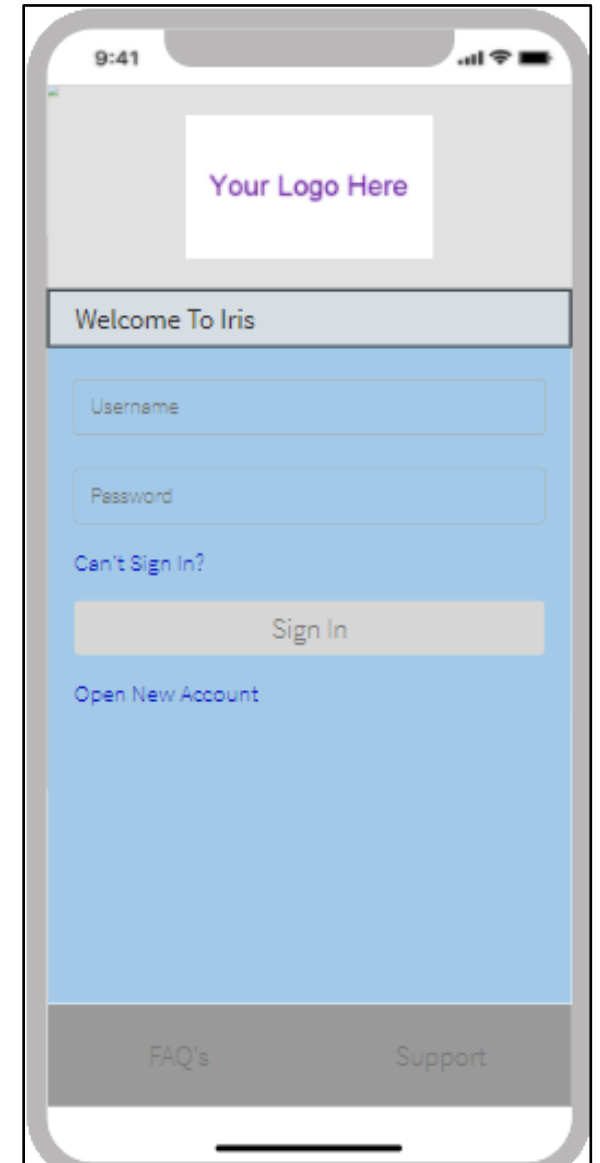
- コントラクトを持つコンポーネントは、以下を持ちます。
 - 独自の名前空間とクラス名
 - UIはビューファイルで、ビジネスロジックはコントローラファイルで、ビジネスロジックをコントローラファイルに記述
- UI階層全体とビジネスロジックは、親コンテナからアクセス可能です。
 - `this.view.parent` は、コントローラ内の親ビューへのアクセスを提供しませんが、親コントローラの `widget.parent` は、その親ビューを提供します。
- コントラクトを持つコンポーネントは、以下のことが可能です。
 - あるプロジェクトからエクスポートして、別のプロジェクトにインポートできます。

ユースケースの定義

- ログイン画面
 - 複数のアプリケーションで利用できるログイン画面を作成します。
 - 認証にはUser Directory Identity Serviceを使用します。
 - 画面を作成し、設定用のプロパティを公開します。

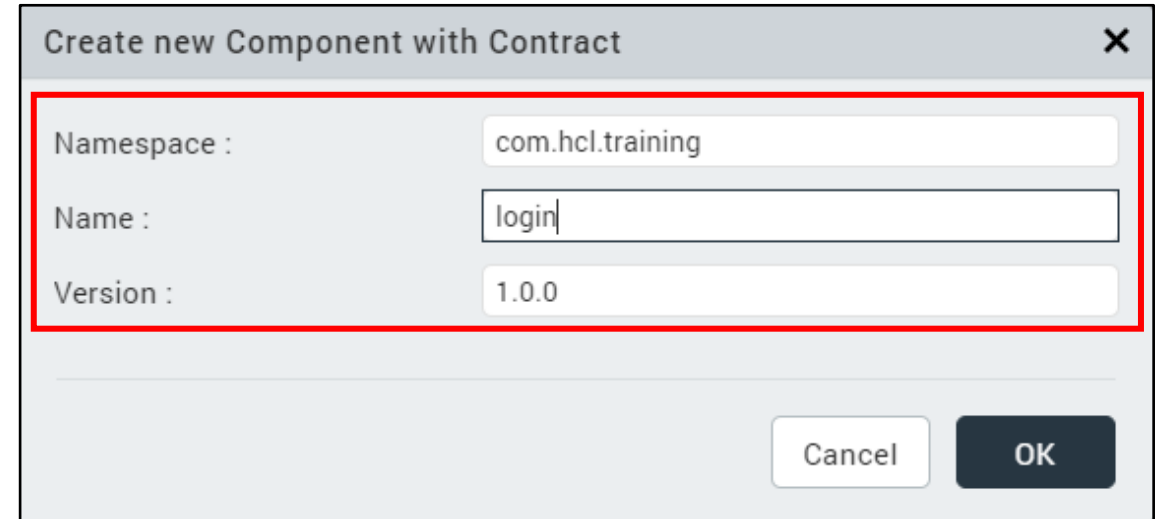
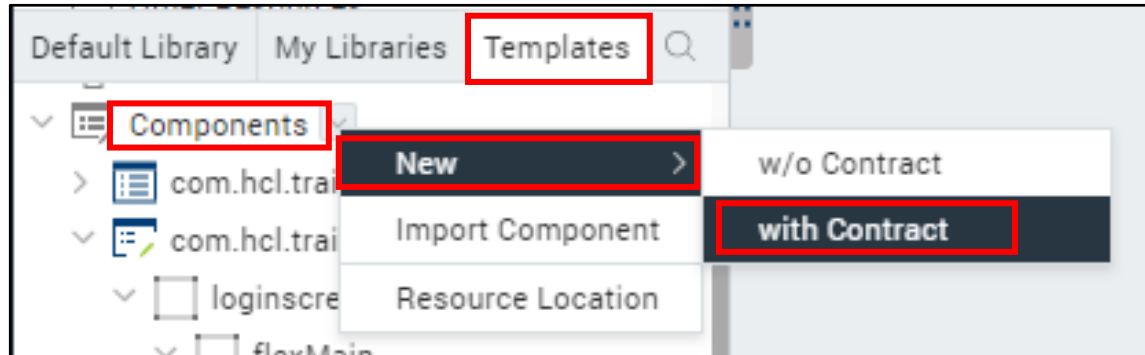
ユースケースの定義（つづき）

- 画面上の以下のリストの要素の可視性、寸法、位置、テキスト、色、フォントファミリー、フォントサイズを設定します。
 - ユーザー名テキストボックス
 - パスワードテキストボックス
 - サインインボタン
 - サインインできないオプション
 - アカウント作成オプション
 - フッターのオプション
- ログイン成功時のコールバックの定義
- ログイン失敗時のコールバックの定義
- ユーザー名とパスワードを検証するために使用する関数を定義する。



新しいプロジェクトの作成

- New project を作成します。
- Templatesセクションに移動し、Componentsをクリックします。
- Namespace, Name に値を入れて OK をクリックします。



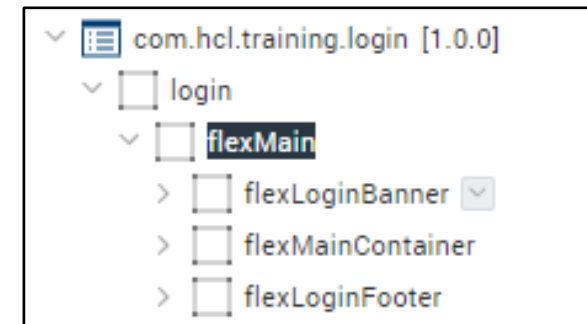
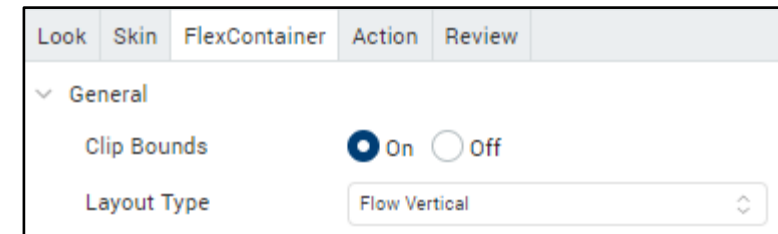
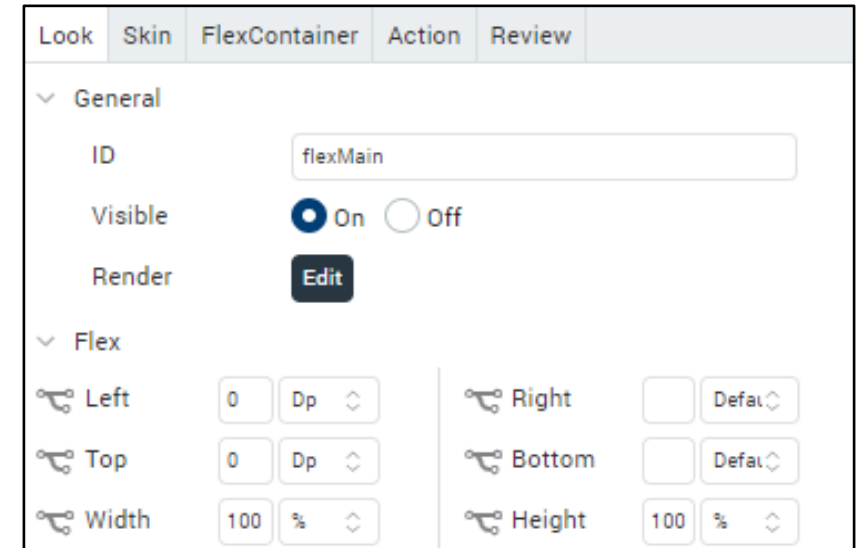
新しいプロジェクトの作成（つづき）

- 以下は、コンポーネントが作成される様子です。
 - コンポーネントを作成すると、フレックスコンテナ、コントローラファイル、コントローラアクションファイルが作成されます。



コンポーネントUIの作成

- フレックスコンテナをコンポーネント内に'flexMain'として追加します。
 - 位置は、左0dp、上0dpです。
 - 寸法は、高さ100%、幅100%で。
- このフレックスコンテナのレイアウトタイププロパティの値を Flow Vertical とします。
- flexMainに、3つのFlexContainerを追加します。
 - flexLoginBanner
 - flexMainContainer
 - flexLoginFooter
- プレビューは次のスライドを参照してください。



コンポーネントUIの作成（続き）

フレックスログインバナーを高さ20%、フリーフォームレイアウトタイプで作成します。

Look	Skin	FlexContainer	Action	Review
<div>General</div> <div>ID flexLoginBanner</div> <div>Visible <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Render <button>Edit</button></div>				
<div>Flex</div> <div>Left 0 Dp Right Default</div> <div>Top 0 Dp Bottom Default</div> <div>Width 100 % Height 20 %</div>				

Look	Skin	FlexContainer	Action	Review
<div>General</div> <div>Clip Bounds <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Layout Type Free Form</div> <div>Modal Container <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Enable Blur <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Enable Haptic Feedback <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Snap to Grid <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Snap Grid Size 10</div>				

フレックスメインコンテナの高さを70%に、レイアウトタイプをフローバーティカルにします。

Look	Skin	FlexContainer	Action	Review
<div>General</div> <div>ID flexMainContainer</div> <div>Visible <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Render <button>Edit</button></div>				
<div>Flex</div> <div>Left 0 Dp Right Default</div> <div>Top 0 Dp Bottom Default</div> <div>Width 100 % Height 70 %</div>				

Look	Skin	FlexContainer	Action	Review
<div>General</div> <div>Clip Bounds <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Layout Type Flow Vertical</div> <div>Child Widget Align Top to Bottom</div> <div>Modal Container <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Enable Blur <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Enable Haptic Feedback <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Snap to Grid <input checked="" type="radio"/> On <input type="radio"/> Off</div>				

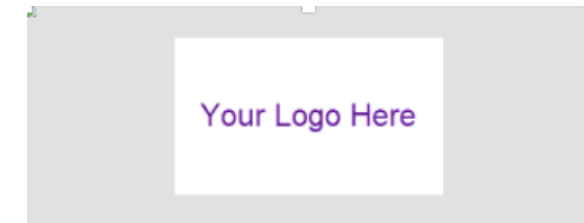
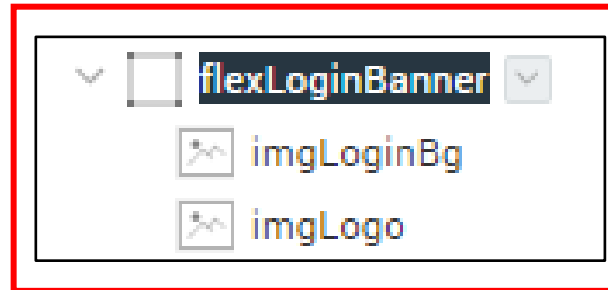
フレックスログインフッターの高さを10%に設定し、フロー横レイアウトタイプにします。

Look	Skin	FlexContainer	Action	Review
<div>General</div> <div>ID flexLoginFooter</div> <div>Visible <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Render <button>Edit</button></div>				
<div>Flex</div> <div>Left 0 Dp Right Default</div> <div>Top 0 Dp Bottom Default</div> <div>Width 100 % Height 10 %</div>				

Look	Skin	FlexContainer	Action	Review
<div>General</div> <div>Clip Bounds <input checked="" type="radio"/> On <input type="radio"/> Off</div> <div>Layout Type Flow Horizontal</div> <div>Child Widget Align Left to Right</div> <div>Modal Container <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Enable Blur <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Enable Haptic Feedback <input type="radio"/> On <input checked="" type="radio"/> Off</div> <div>Snap to Grid <input checked="" type="radio"/> On <input type="radio"/> Off</div>				

コンポーネントUIの作成（続き）

- flexLoginBannerを構築し、2つの画像を追加します。

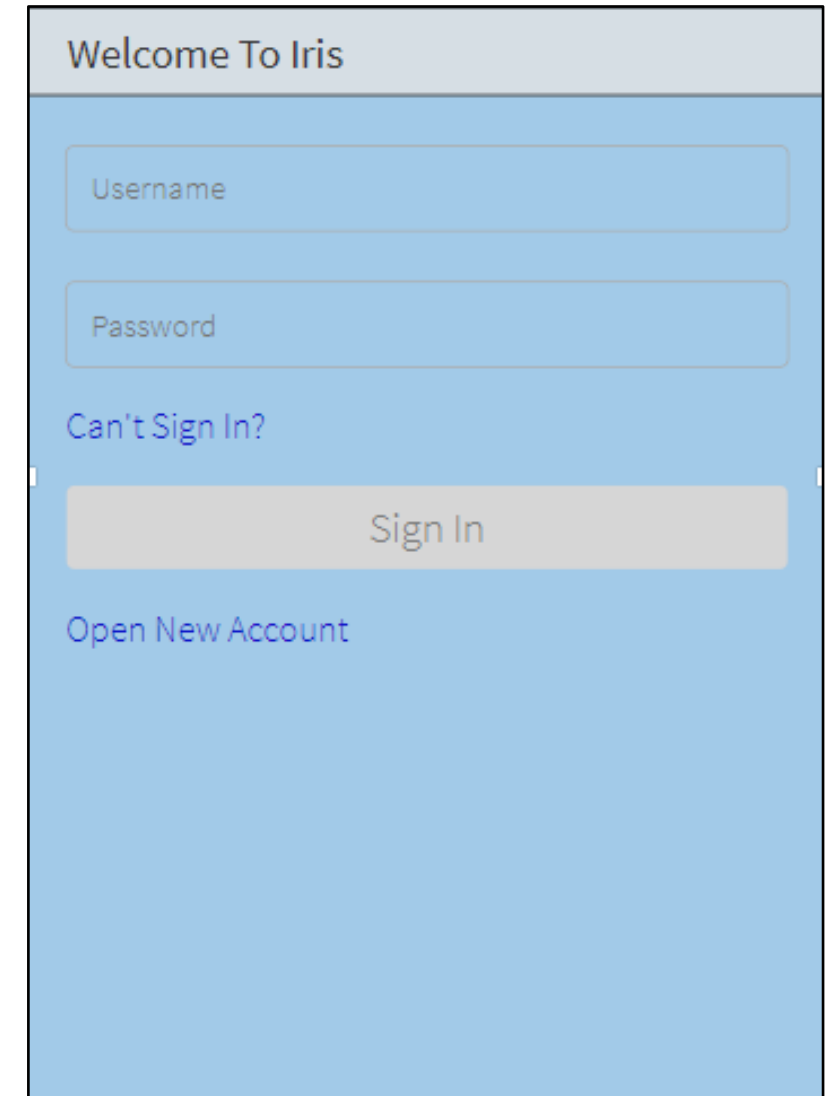
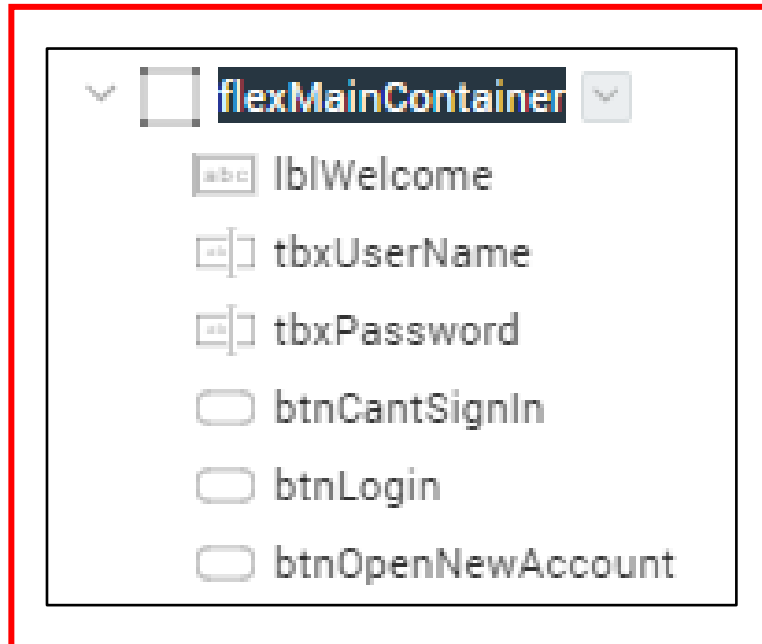


Look	Skin	Image	Action	Review
▼ General				
ID	imgLoginBg			
Visible	<input checked="" type="radio"/> On <input type="radio"/> Off			
Render	<button>Edit</button>			
▼ Flex				
Left	<input type="text"/>	Defal	Right	<input type="text"/> Defal
Top	<input type="text"/>	Defal	Bottom	<input type="text"/> Defal
Width	100	%	Height	100 %
Min Width	<input type="text"/>	Defal	Max Width	<input type="text"/> Defal
Min Height	<input type="text"/>	Defal	Max Height	<input type="text"/> Defal
Center X	50	%	Center Y	50 %

Look	Skin	Image	Action	Review
▼ General				
ID	imgLogo			
Visible	<input checked="" type="radio"/> On <input type="radio"/> Off			
Render	<button>Edit</button>			
▼ Flex				
Left	<input type="text"/>	Defal	Right	<input type="text"/> Defal
Top	<input type="text"/>	Defal	Bottom	<input type="text"/> Defal
Width	200	Dp	Height	70 %
Min Width	<input type="text"/>	Defal	Max Width	<input type="text"/> Defal
Min Height	<input type="text"/>	Defal	Max Height	<input type="text"/> Defal
Center X	50	%	Center Y	50 %

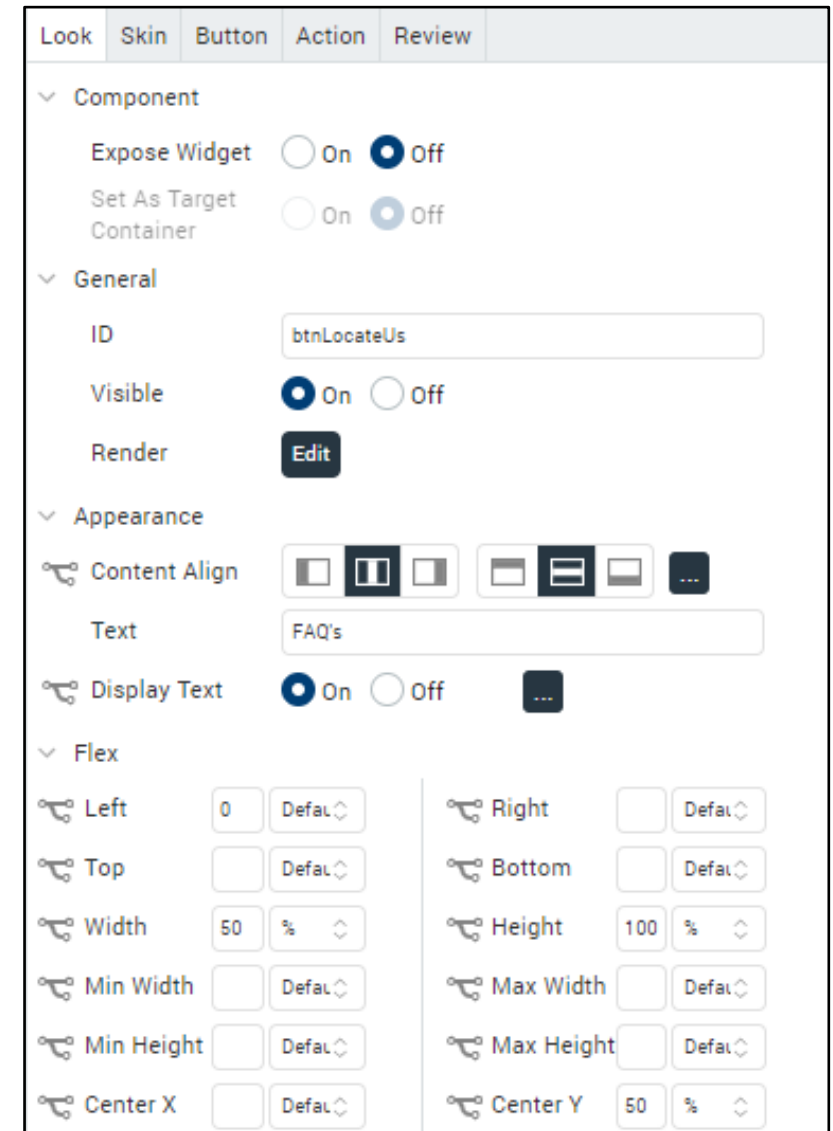
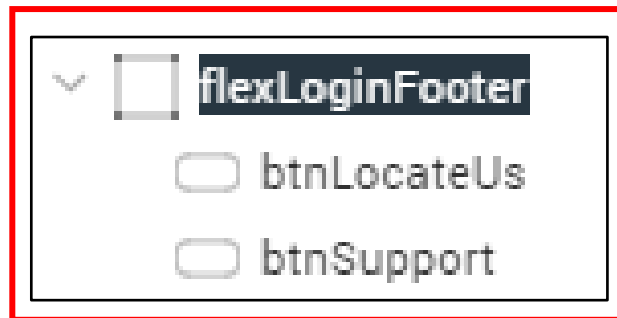
コンポーネントUIの作成（続き）

- flexMainContainerを構築して、図のウィジェットを追加します。



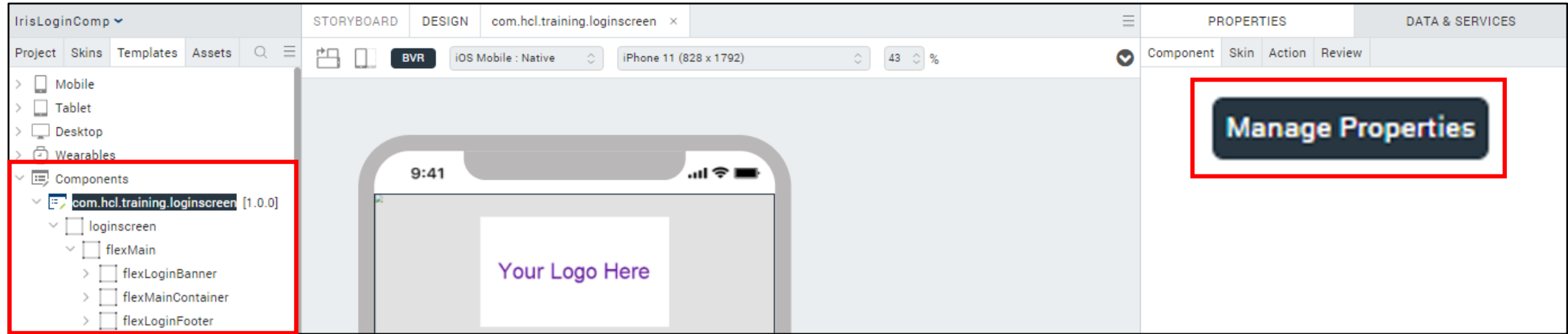
コンポーネントUIの作成（続き）

- flexLoginFooterを構築して、以下のウィジェットを追加します。
- このフレックスコンテナはFlow Horizontalに設定されているので、どちらのボタンも同じ外観のプロパティを持っています。



プロパティの公開

- コンポーネント(com.hcl.training.login)を選択し、コンポーネントタブを選択します。

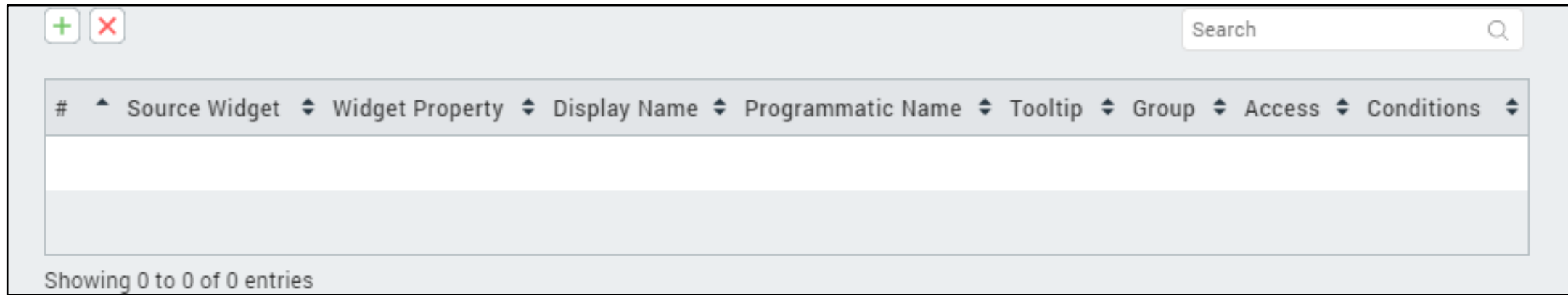


- 公開できるプロパティは2種類あります。

パススループロパティ	カスタムプロパティ
<ul style="list-style-type: none"> パススループロパティは、プラットフォームの一部として利用可能なウィジェットプロパティです。 パススループロパティの値が変更されたときに、プラットフォームがどのように反応するかが分かります。 	<ul style="list-style-type: none"> カスタムプロパティは、コンポーネント作成者によって定義されるプロパティです。 コンポーネント作成者は、カスタムプロパティの値が変更されたときに何が起こるかについてのロジックを実装する必要があります。

パススループロパティの公開

- パススループロパティは、コンポーネントの使用者に公開したいプラットフォームの一部として利用可能なウィジェットプロパティです。



Source Widget	公開したいプロパティを持つウィジェットを選択します。
Widget Property	公開するウィジェットのプロパティを選択します。
Display Name	このプロパティを公開するための名前を定義します。
Programmatic Name	使用者がコードからこのプロパティの値にアクセスし、変更できるようにするための、プロパティのプログラム名を定義します。
Group	プロパティのグループを作成できます。Flex', 'General', 'Background', 'Border'などのグループと同じです。
Access	プロパティを有効または無効にできます。

パススループロパティの公開

- 以下のプロパティを公開します。
- グループ ドロップダウンを使用して、グループ化を作成および管理します。

1	flexMain	skin	mainFlexSkin	mainFlexSkin	General	Enable	Conditions
2	flexMainContainer	skin	loginCardSkin	loginCardSkin	General	Enable	Conditions
3	imgLoginBg	src	logoBackgroundImage	logoBackgroundImage	Logo Properties	Enable	Conditions
4	imgLogo	src	logoImageSource	logoImageSource	Logo Properties	Enable	Conditions
5	lblWelcome	skin	welcomeLabelSkin	welcomeLabelSkin	Welcome Message	Enable	Conditions
6	lblWelcome	text	welcomeLabelText	welcomeLabelText	Welcome Message	Enable	Conditions
7	tbxUserName	skin	userNameTextBoxSkin	userNameTextBoxSkin	Username Properties	Enable	Conditions
8	tbxUserName	placeholder	userNameTextBoxPlaceholder	userNameTextBoxPlaceholder	Username Properties	Enable	Conditions
9	tbxUserName	text	userNameTextBoxText	userNameTextBoxText	Username Properties	Enable	Conditions
10	tbxUserName	keyBoardStyle	userNameKeyboardStyle	userNameKeyboardStyle	Username Properties	Enable	Conditions
11	tbxPassword	skin	passwordTextBoxSkin	passwordTextBoxSkin	Password Properties	Enable	Conditions
12	tbxPassword	text	passwordTextBoxText	passwordTextBoxText	Password Properties	Enable	Conditions
13	tbxPassword	placeholder	passwordTextBoxPlaceholder	passwordTextBoxPlaceholder	Password Properties	Enable	Conditions
14	tbxPassword	keyBoardStyle	passwordTextBoxKeyboardSt	passwordTextBoxKeyBoardSt	Password Properties	Enable	Conditions
15	tbxPassword	secureTextEntry	passwordTextBoxSecureText	passwordTextBoxSecureText	Password Properties	Enable	Conditions
16	btnCantSignIn	skin	cantSignInButtonSkin	cantSignInButtonSkin	Can't Signin Properties	Enable	Conditions
17	btnCantSignIn	focusSkin	cantSignInButtonFocusSkin	cantSignInButtonFocusSkin	Can't Signin Properties	Enable	Conditions
18	btnCantSignIn	text	cantSignInButtonText	cantSignInButtonText	Can't Signin Properties	Enable	Conditions

パススループロパティの公開

- 以下は、公開する残りのプロパティです。

19	btnLogin	skin	loginButtonSkin	Skin	Login Button Properties	Enable	Conditions
20	btnLogin	focusSkin	loginButtonFocusSkin	loginButtonFocusSkin	Login Button Properties	Enable	Conditions
21	btnLogin	text	loginButtonText	loginButtonText	Login Button Properties	Enable	Conditions
22	btnOpenNewAccount	skin	openNewButtonSkin	openNewButtonSkin	Open New Account Proper	Enable	Conditions
23	btnOpenNewAccount	focusSkin	openNewButtonFocusSkin	openNewButtonFocusSkin	Open New Account Proper	Enable	Conditions
24	btnOpenNewAccount	text	openNewButtonText	openNewButtontext	Open New Account Proper	Enable	Conditions
25	btnLocateUs	skin	footerLocateUsNormal	footerLocateUsNormal	Footer Options Properties	Enable	Conditions
26	btnLocateUs	focusSkin	footerLocateUsFocusSkin	footerLocateUsFocusSkin	Footer Options Properties	Enable	Conditions
27	btnLocateUs	text	footerLocateUsText	footerLocateUsText	Footer Options Properties	Enable	Conditions
28	btnSupport	skin	footerSupportSkin	footerSupportSkin	Footer Options Properties	Enable	Conditions
29	btnSupport	focusSkin	footerSupportFocusSkin	footerSupportFocusSkin	Footer Options Properties	Enable	Conditions
30	btnSupport	text	footerSupportText	footerSupportText	Footer Options Properties	Enable	Conditions
31	flexLoginFooter	skin	footerSkin	footerSkin	Footer Options Properties	Enable	Conditions

カスタムプロパティの公開

- カスタムプロパティは、どのウィジェットにも関連付けられていません。これらは、コンポーネントの作成者によって定義されます。
- このようなカスタムプロパティの値が変更されたときに、コンポーネントがどのように動作するかは、コンポーネント作成者が定義する必要があります。

#	Property Name	Display Name	Tooltip	Property Type	Value	Default Value	Group	Read / Write	Conditions
1	loginFailureMessage	loginFailureMessage		String		Login Failed!	Login Messages	Write	Conditions
2	loginSuccessMessage	loginSuccessMessage		String		Login Success!	Login Messages	Write	Conditions

Property Name	カスタムプロパティの名前を定義します。
Display Name	カスタムプロパティがIrisのプロパティパネルにどのように表示されるかを定義します。
Property Type	カスタムプロパティに設定できる値の型を定義します。例えば、String、Integer、Booleanなど。
Default Value	カスタムプロパティのデフォルト値を定義します。
Group	カスタムプロパティのグループを作成できます。
Read / Write	プロパティが読み取り専用か読み取り/書き込み可能かを指定します。

パススループロパティの公開（続き）

- 完成すると、前のスライドのステップに従って作成したプロパティになります。

The screenshot shows the 'Manage Properties' dialog box with the 'Pass Through' tab selected. The table lists 9 properties, each with a source widget, widget property, display name, programmatic name, tooltip, group, access, and conditions. The 'Group' column is highlighted with a red box.

#	Source Widget	Widget Property	Display Name	Programmatic Name	Tooltip	Group	Access	Conditions
1	flexMain	skin	mainFlexSkin	mainFlexSkin		General	Enable	Conditions
2	flexMainContaine	skin	loginCardSkin	loginCardSkin		General	Enable	Conditions
3	imgLoginBg	src	logoBackgroundImage	logoBackgroundImage		Logo Properties	Enable	Conditions
4	imgLogo	src	logoImageSource	logoImageSource		Logo Properties	Enable	Conditions
5	lblWelcome	skin	welcomeLabelSkin	welcomeLabelSkin		Welcome Message	Enable	Conditions
6	lblWelcome	text	welcomeLabelText	welcomeLabelText		Welcome Message	Enable	Conditions
7	tbxUserName	skin	userNameTextBoxSkin	userNameTextBoxSkin		Username Properties	Enable	Conditions
8	tbxUserName	placeholder	userNameTextBoxPlaceholde	userNameTextBoxPlaceholde		Username Properties	Enable	Conditions
9	tbxUserName	text	userNameTextBoxText	userNameTextBoxText		Username Properties	Enable	Conditions

Showing 1 to 31 of 31 entries

Buttons: Cancel, Apply

- ユーザーを支援するためのグループ分けに注目

パススループロパティの公開（続き）

- 完成すると、前のスライドのステップに従って作成したプロパティになります。

Component Skin Action Review

Manage Properties

Logo Properties

logoBackgroundImage Select an Image Edit

logoImageSource logonew.png Edit

Welcome Message

welcomeLabelText Welcome To Iris

Username Properties

userNameTextBoxPlaceholder Username

userNameTextBoxText

userNameKeyboardStyle Default ...

Password Properties

passwordTextBoxText

passwordTextBoxPlaceholder Password

passwordTextBoxKeyboardStyle Default ...

passwordTextBoxSecureText ☒ On ☐ Off

Can't Signin Properties

cantSigninButtonText Can't Sign In?

Login Button Properties

loginButtonText Sign In

Open New Account Properties

openNewButtonText Open New Account

Footer Options Properties

footerLocateUsText FAQ's

footerSupportText Support

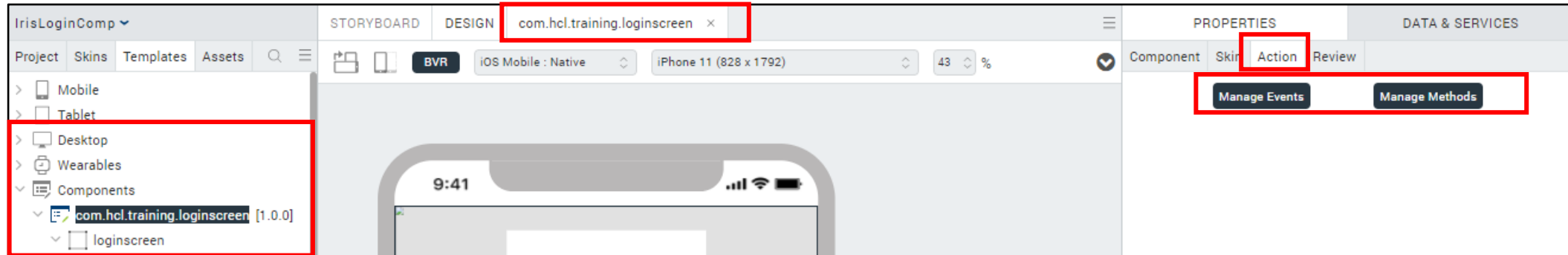
Login Messages

loginFailureMessage Login Failed!

loginSuccessMessage Login Success!

イベントとメソッドの公開

- コンポーネントのイベントとメソッドを公開します。



イベント

- イベントは、ユーザーによってコンポーネント上でアクションが実行されたときにトリガーされます。
 - 例えば、ボタンでは 'onClick' イベントを、リストボックスでは 'onSelection' イベントを実行するなどです。

メソッド

- メソッドを使用すると、コンポーネントの使用者はコンポーネント内のデータにアクセス(読み取りまたは書き込み)できます。
 - 例えば、'setVisibility' メソッドは、任意のウィジェットの可視性を変更できます、等々。

カスタムイベントの公開

- カスタムイベントは、コンポーネント作成者によって定義されます。
- コンポーネント作成者は、これらのイベントをいつ発生させるかも定義する必要があります。
- カスタム・イベントを定義する際には、以下の属性を定義する必要があります。

Manage Events

Pass Through Custom

+ - Manage Events Search

#	Raised Events	Tooltip	Group
1	validatePassword		Validation Events
2	validateUsername		Validation Events
3	loginSuccessCallback		Login Event
4	loginFailureCallback		Login Event

Showing 1 to 4 of 4 entries

Cancel Apply

Raised Events

カスタムイベントの名称を定義します。

Group

カスタムイベントのグループを作成できます。

カスタムイベントの公開（つづき）

- これは、前のスライドで示された定義に対して、カスタムイベントが作成される方法です。

#	Raised Events	Group
1	validatePassword	Validation Events
2	validateUsername	Validation Events
3	loginSuccessCallback	Login Events
4	loginFailureCallback	Login Events

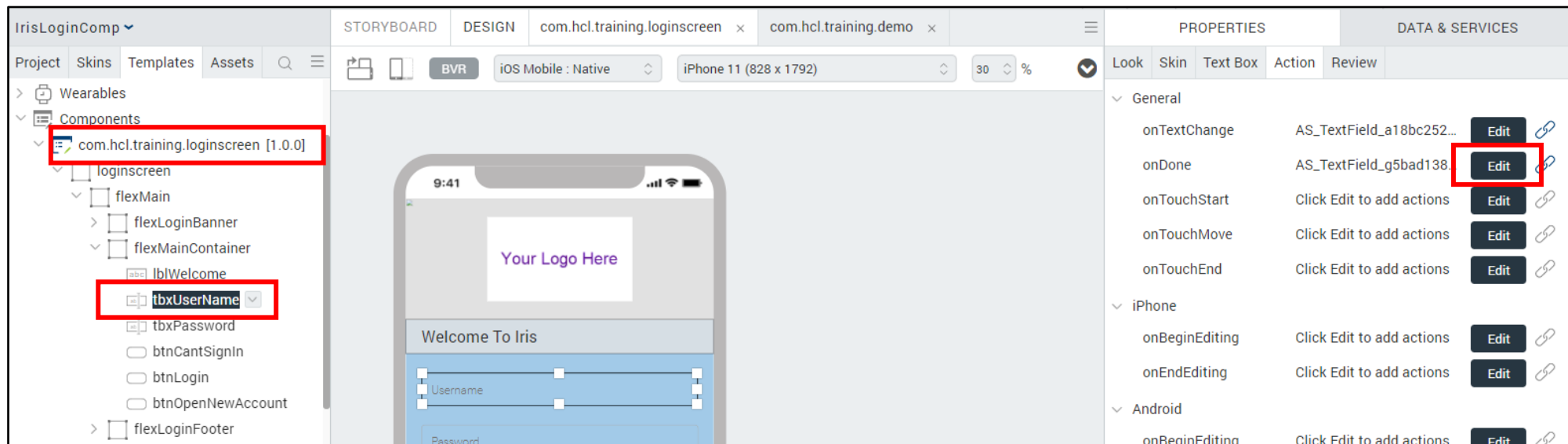
Component Skin Action Review

Manage Events Manage Methods

- Validation Events
 - validatePassword
 - validateUsername
- Login Event
 - loginSuccessCallback
 - loginFailureCallback

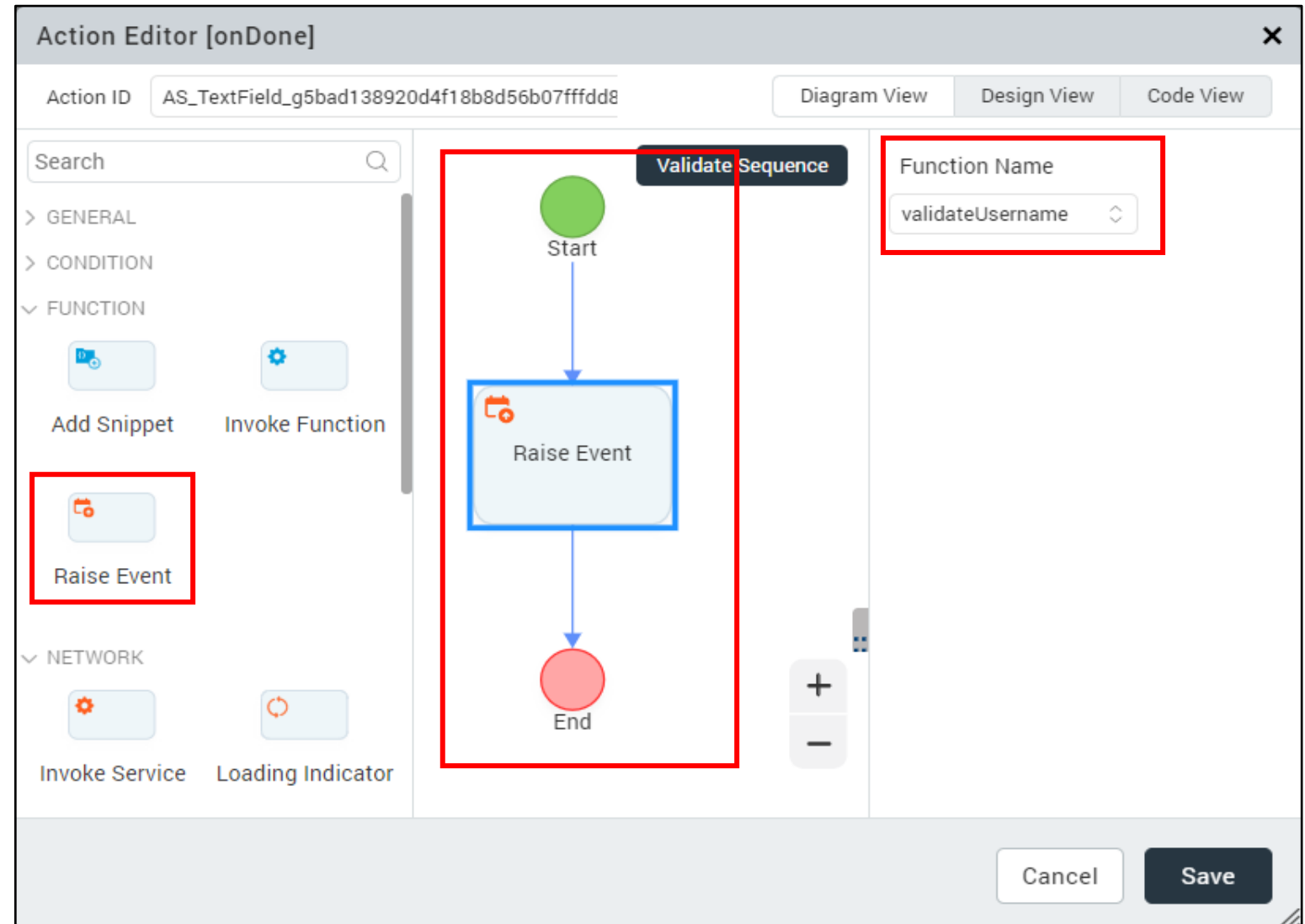
カスタムイベントの公開（つづき）

- カスタムイベント 'validatePassword' と 'validateUsername' は、ユーザー名テキストボックスとパスワードテキストボックスの 'onDone' イベントで発生します。
- 'validateUsername' と 'validatePassword' の実際のアクションシーケンスは、コンポーネントのコンシューマによって定義されます。



カスタムイベントの公開（つづき）

- カスタム・イベントの発生
(Raising Event)
 - コンポーネントのコンシューマがカスタムイベントに対して定義したアクションシーケンスは、図のような形で発生します。



カスタムメソッドの公開

- カスタムメソッドは、コンポーネントの作成者によって定義されます。
- コンポーネント作成者は、これらのカスタムメソッドのロジックも定義する必要があります。
- カスタムメソッドを定義する際には、以下の属性を定義する必要があります。

Manage Methods

Pass Through **Custom**

+ - Manage Methods Search

#	Method	Group
1	getUsername	General

Showing 1 to 1 of 1 entries

Cancel Apply

Method	カスタムメソッドの名前を定義する
Group	カスタムメソッド用のグループを作成できます。

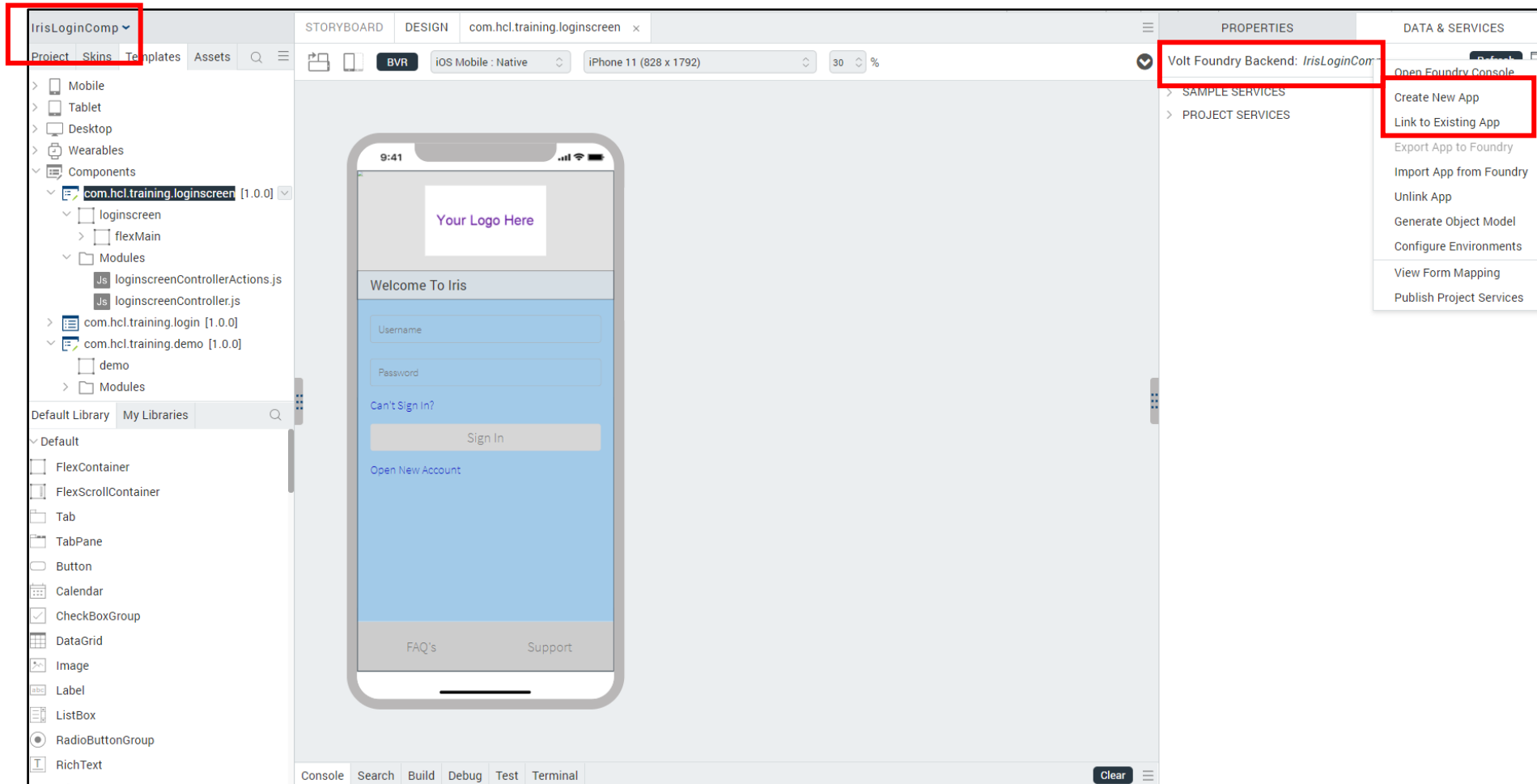
カスタムメソッドの公開

- 以下のコントローラロジックを確認します。
 - カスタムプロパティのinitGetterSettersを更新します。
 - カスタムメソッド用のロジックを追加します。

```
//Logic for getters/setters of custom properties
initGetterSetters: function() {
  defineGetter(this, 'loginFailureMessage', () => {
    return this._loginFailureMessage;
  });
  defineSetter(this, 'loginFailureMessage', value => {
    this._loginFailureMessage = value;
  });
  defineGetter(this, 'loginSuccessMessage', () => {
    return this._loginSuccessMessage;
  });
  defineSetter(this, 'loginSuccessMessage', value => {
    this._loginSuccessMessage = value;
  });
},
getUsername: function() {
  return this.view.tbxUserName.text;
}
```

コンポーネントをFoundryにリンクする

- Iris プロジェクトを Foundry プロジェクトにリンクするか、Identity Service を持つ新しいプロジェクトを作成します。



コンポーネントをFoundryにリンクする（続き）

- Identity サービスのタイプは User Repository です。

The screenshot displays the HCL Foundry console interface. At the top, the user is logged in as Seth Weissman. The main navigation bar includes tabs for 'Configure Services' and 'Manage Client App Assets'. Under 'Configure Services', the 'Identity' tab is selected and highlighted with a red box. Below the navigation bar, there are three buttons: 'CONFIGURE NEW', 'USE EXISTING', and 'SERVICE CONFIGURATION'. A search bar is also present. The main content area shows a table of configured services. The first row is highlighted with a red box, showing the 'IrisLoginIdentitySvc' service, which is a 'User Repository' type, with a URL of 'https://100000037.auth.d...', SSO status 'Disabled', modified by 'Seth Weissman', and on '14 May 2022 16:31 UTC'.

	NAME	URL	TYPE	SSO	MODIFIED BY	MODIFIED ON
<input type="checkbox"/>	IrisLoginIdentitySvc	https://100000037.auth.d...	User Repository	Disabled	Seth Weissman	14 May 2022 16:31 UTC

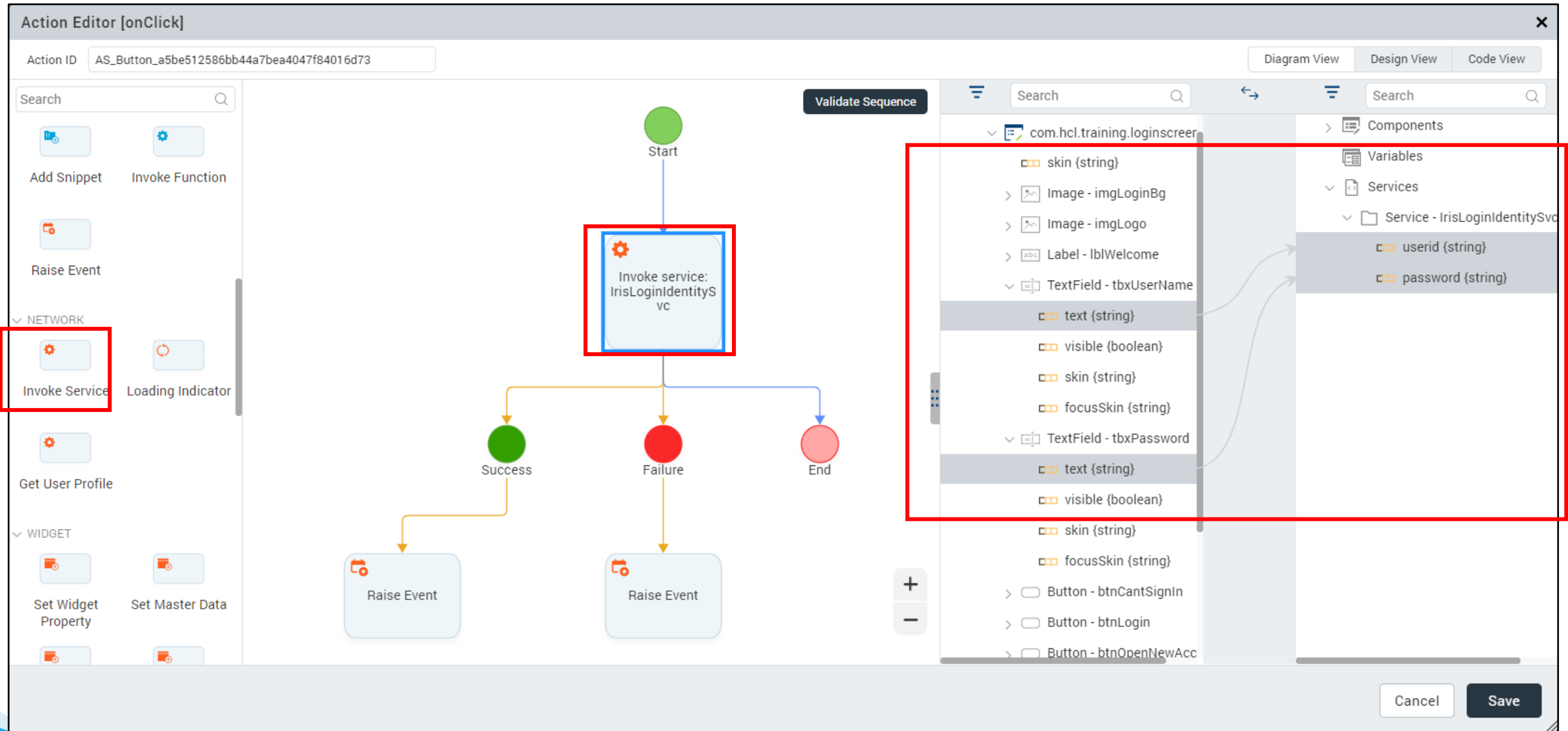
コンポーネントをFoundryにリンクする（続き）

The screenshot displays the HCL Foundry IDE interface. On the left, the 'Components' panel shows a tree structure for 'com.hcl.training.loginscreen [1.0.0]'. The 'btnLogin' component is highlighted with a red box. Below it, the 'Default Library' panel lists various UI components like FlexContainer, FlexScrollContainer, Tab, TabPane, Button, Calendar, CheckBoxGroup, DataGrid, Image, Label, ListBox, RadioButtonGroup, and RichText.

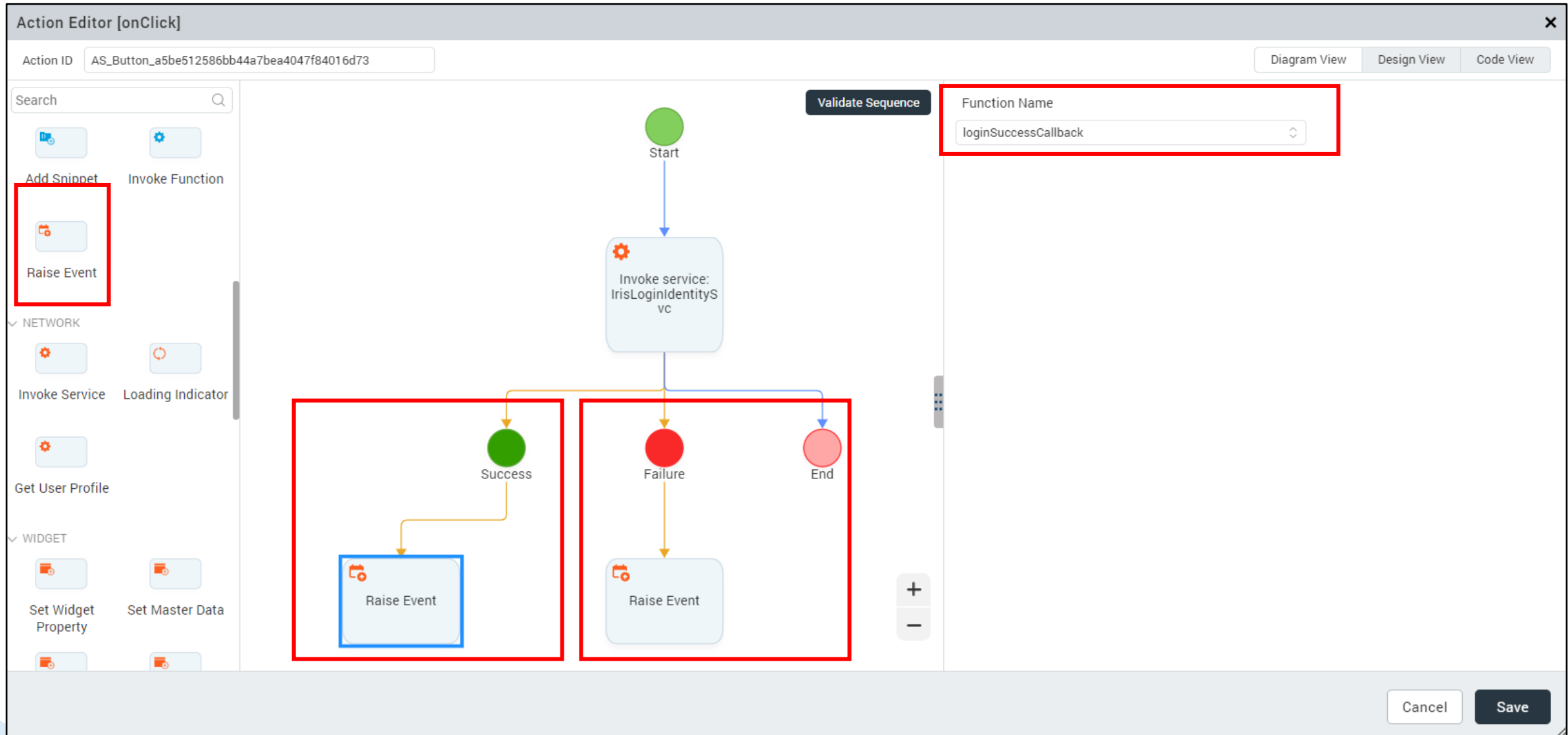
The central canvas shows a mobile app design for an iPhone 11 (828 x 1792). The app has a header 'Your Logo Here', a title 'Welcome To Iris', and two input fields for 'Username' and 'Password'. Below these is a 'Sign In' button, which is highlighted with a red box. There are also links for 'Can't Sign In?' and 'Open New Account'.

On the right, the 'PROPERTIES' panel is open, showing the 'General' tab. The 'onClick' action is highlighted with a red box, and its value is 'AS_Button_a5be512586...'. The 'Edit' button next to it is also highlighted with a red box.

コンポーネントをFoundryにリンクする（続き）



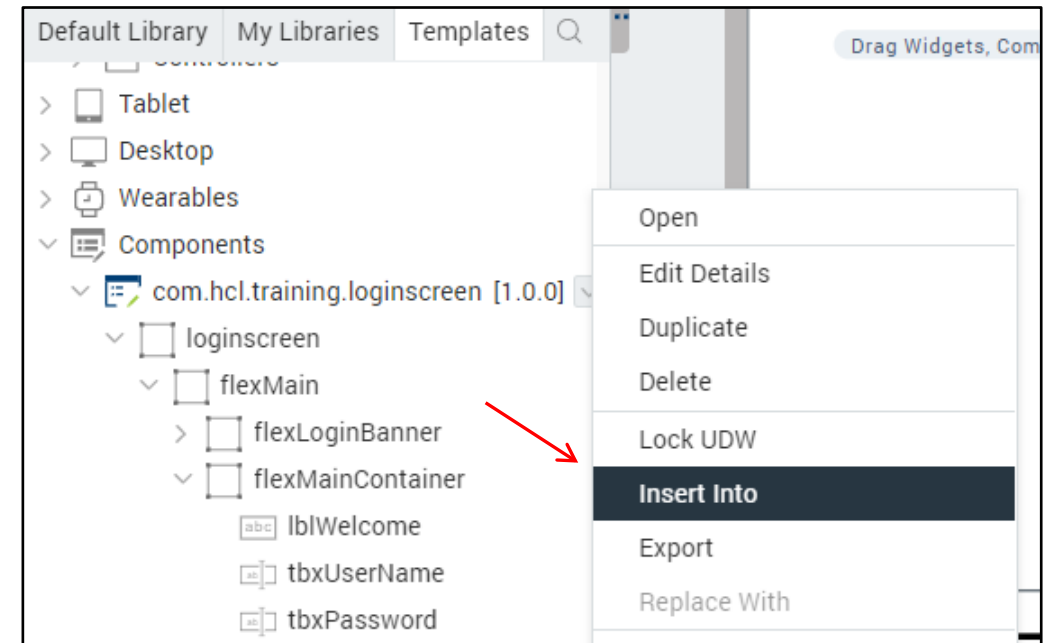
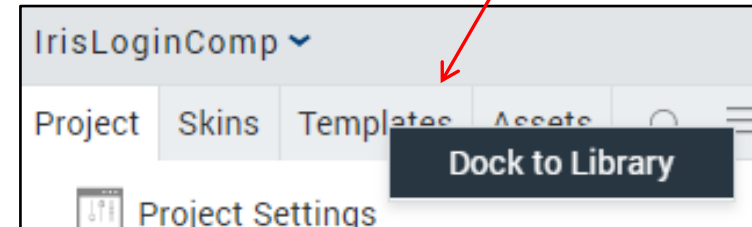
コンポーネントをFoundryにリンクする（続き）



フォームでコンポーネントを使用する

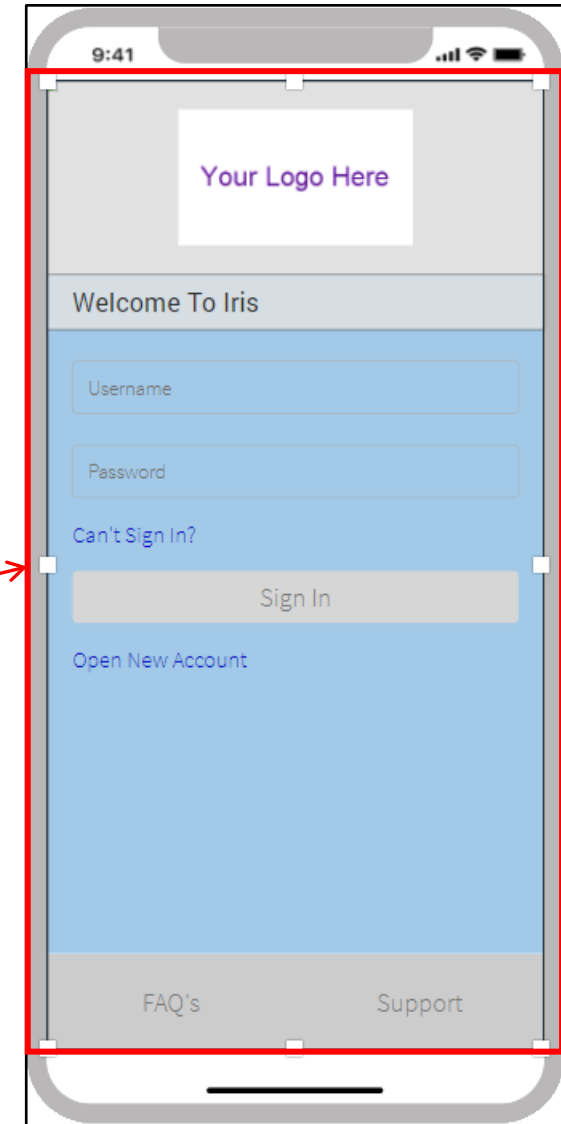
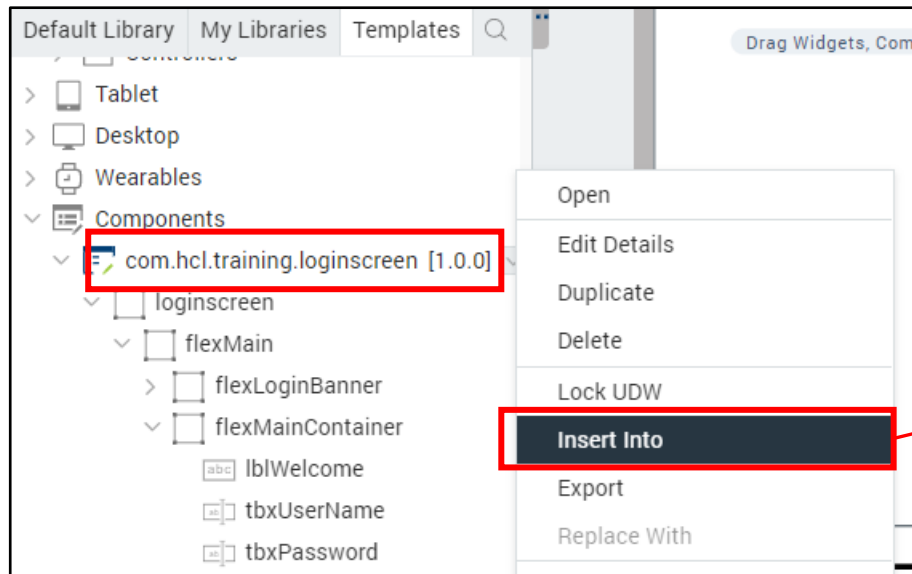
- フォームを開く
- Template タブを開く
- ヒント:「テンプレート」タブを「ライブラリー」エリアに移動する
- コンポーネントを右クリック（またはコンテキストメニューから）-“insert into” オプションを選択。
- コンポーネントは、フォームに追加されます。
- または、フォームにドラッグすることもできます。

ヒント: アプリにコンポーネントを簡単に追加するには、テンプレートタブをライブラリーに移動します。



フォームでコンポーネントを使用する（続き）

- コンポーネントを追加した後のログイン画面は以下の通りです。



コンポーネントのカスタマイズ

The screenshot displays the HCL IDE interface for customizing a login screen component. The Project Explorer on the left shows the project structure, with 'Form2' selected under 'com.hcl.training.loginscreen'. The central storyboard view shows a mobile app design with a login form. The Properties panel on the right, titled 'Component', lists various properties for the selected component, including ID, Logo Properties, Welcome Message, Username Properties, Password Properties, Can't Signin Properties, Login Button Properties, Open New Account Properties, and Footer Options Properties. The 'Form2' component is highlighted in the Project Explorer, and its properties are being edited in the Properties panel.

Project Explorer:

- Project Settings
- Mobile
 - Splash Screen
 - Forms
 - CopyForm2
 - Form1
 - Form3
 - Form2**
 - loginscreen
 - Controllers
- Tablet
- Responsive Web / Desktop
- Wearables
- Modules
- Web

Properties Panel:

- ID: loginscreen
- Logo Properties
 - logoBackgroundImage: Select an Image (Edit)
 - logoImageSource: logonew.png (Edit)
- Welcome Message
 - welcomeLabelText: Welcome To Iris
- Username Properties
 - userNameTextBoxPlaceholder: Username
 - userNameTextBoxText:
 - userNameKeyboardStyle: Default
- Password Properties
 - passwordTextBoxText:
 - passwordTextBoxPlaceholder: Password
 - passwordTextBoxKeyboardStyle: Default
 - passwordTextBoxSecureText: ☒ On ☐ Off
- Can't Signin Properties
 - cantSigninButtonText: Can't Sign In?
- Login Button Properties
 - loginButtonText: Sign In
- Open New Account Properties
 - openNewButtonText: Open New Account
- Footer Options Properties
 - footerLocateUsText: FAQ's
 - footerSupportText: Support

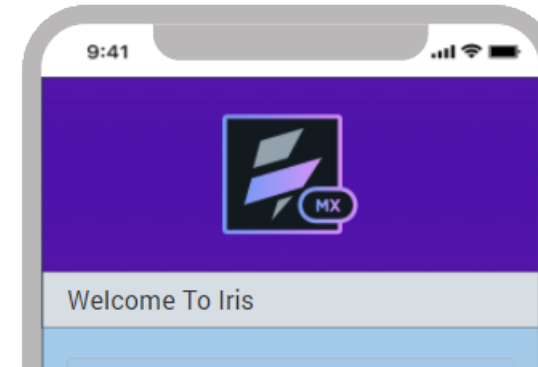
コンポーネントのカスタマイズ（続き）

The screenshot displays the HCL Iris IDE interface for customizing a mobile application component. The left sidebar shows the project structure, with the 'loginscreen' component selected under 'Forms'. The central canvas shows a mobile app preview with a login form. The right-hand 'PROPERTIES' panel is open to the 'Action' tab, which is highlighted with a red box. This tab lists various events like 'onTouchStart', 'validatePassword', and 'loginSuccessCallback', each with an 'Edit' button. A red rectangle also highlights the 'loginscreen' component in the project tree and the 'Action' tab in the properties panel.

Component	Look	Skinnable	Action	Review
General				
onTouchStart			Click Edit to add actions	Edit
onTouchMove			Click Edit to add actions	Edit
onTouchEnd			Click Edit to add actions	Edit
Validation Events				
validatePassword			Click Edit to add actions	Edit
validateUsername			Click Edit to add actions	Edit
Login Event				
loginSuccessCallback			Click Edit to add actions	Edit
loginFailureCallback			Click Edit to add actions	Edit

コンポーネントのカスタマイズ（続き）

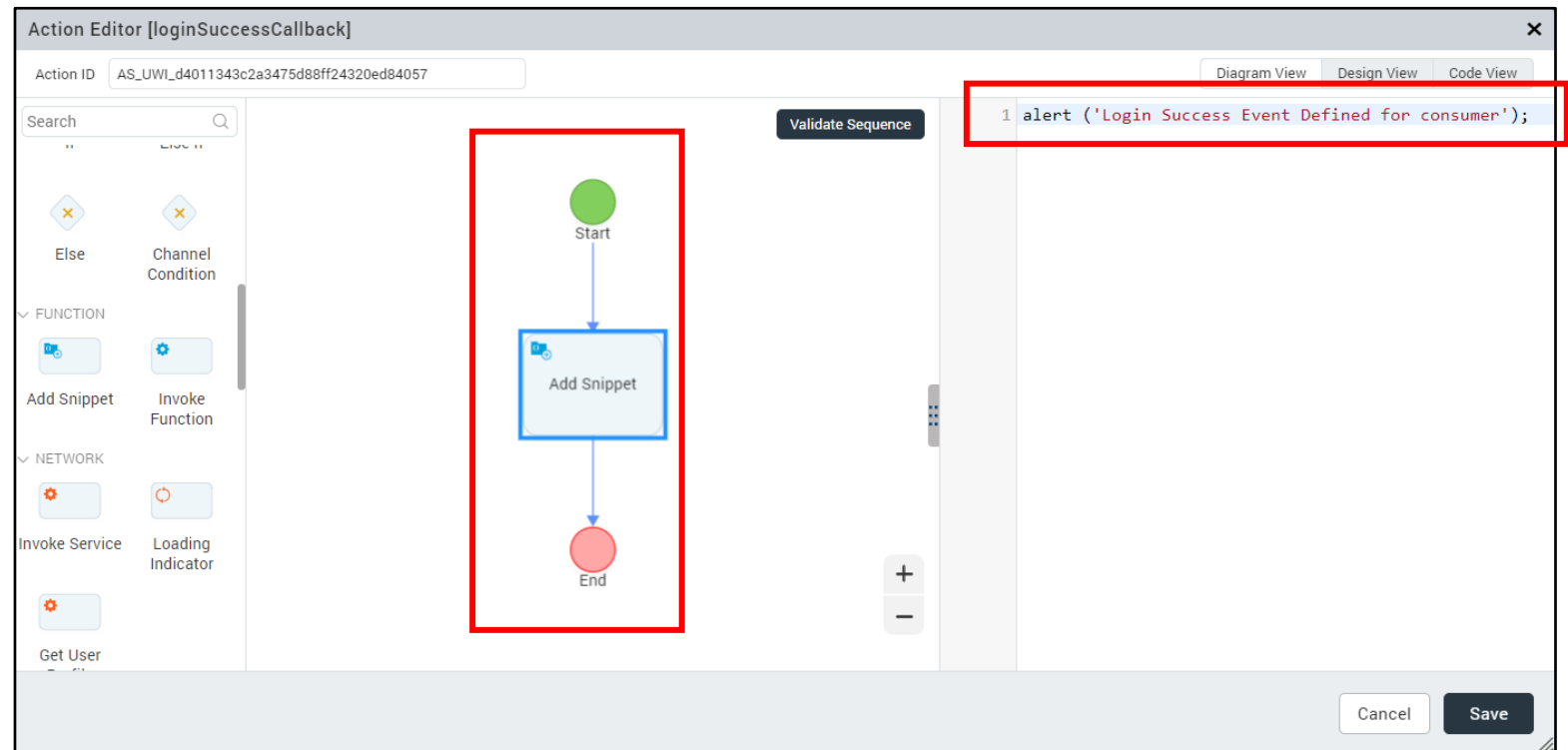
PROPERTIES		DATA & SERVICES	
Component	Look	Skin	Action
ID	loginscreen		
▼ Logo Properties			
logoBackgroundImage	purplebg.png	<button>Edit</button>	
logoImageSource	voltmx_150.png	<button>Edit</button>	



PROPERTIES		DATA & SERVICES	
Component	Look	Skin	Action
Exposed Skins			
None			
cantSignInButtonFocusSkin (cantSignInButtonFocusSkin)			
cantSignInButtonSkin (cantSignInButtonSkin)			
footerLocateUsFocusSkin (footerLocateUsFocusSkin)			
footerLocateUsNormal (footerLocateUsNormal)			
footerSkin (footerSkin)			
footerSupportFocusSkin (footerSupportFocusSkin)			
footerSupportSkin (footerSupportSkin)			
loginButtonFocusSkin (loginButtonFocusSkin)			
loginButtonSkin (Skin)			
loginCardSkin (loginCardSkin)			
mainFlexSkin (mainFlexSkin)			
openNewButtonFocusSkin (openNewButtonFocusSkin)			
openNewButtonSkin (openNewButtonSkin)			
passwordTextBoxSkin (passwordTextBoxSkin)			
userNameTextBoxSkin (userNameTextBoxSkin)			
welcomeLabelSkin (welcomeLabelSkin)			

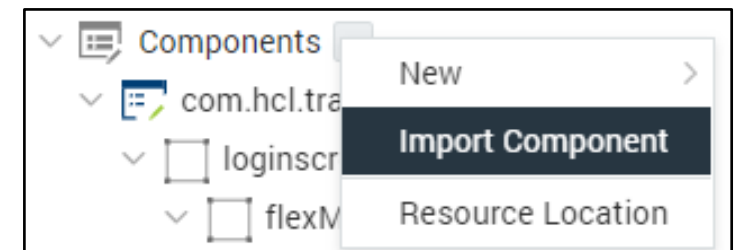
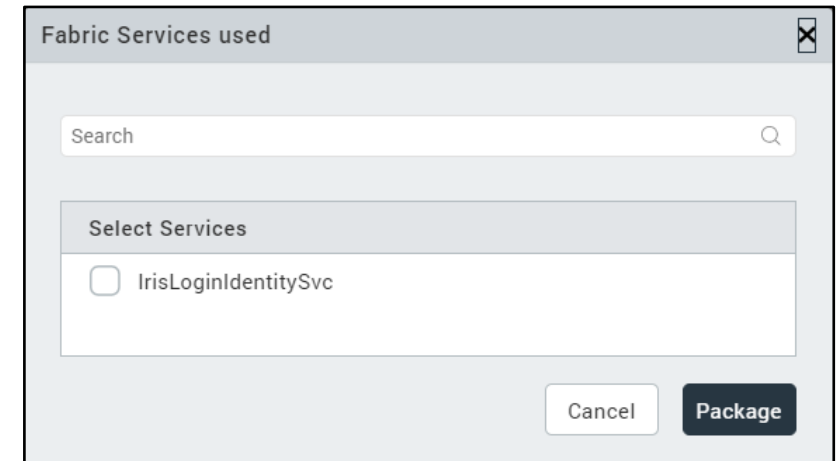
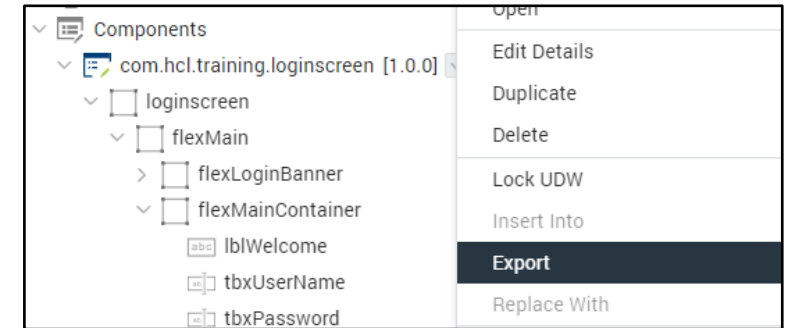
コンポーネントのカスタマイズ（続き）

PROPERTIES			DATA & SERVICES	
Component	Look	Skin	Action	Review
General				
onTouchStart			Click Edit to add actions	<button>Edit</button>
onTouchMove			Click Edit to add actions	<button>Edit</button>
onTouchEnd			Click Edit to add actions	<button>Edit</button>
Validation Events				
validatePassword			Click Edit to add actions	<button>Edit</button>
validateUsername			Click Edit to add actions	<button>Edit</button>
Login Event				
loginSuccessCallback			Click Edit to add actions	<button>Edit</button>
loginFailureCallback			Click Edit to add actions	<button>Edit</button>



コンポーネントのエクスポートとインポート

- エクスポートとインポートによるバックアップと共有できます。
- エクスポートする際に、どのサービスを含めるかを選択します。
- インポートオプションを選択して、コンポーネントを追加します。



練習

- このレッスンで説明した手順に従ってください。
- 設定するプロパティを追加する。
- 別のプロジェクトで使用するためにエクスポートとインポートを行う。

HCL

www.hcltech.com

\$10 BILLION | 159,000+ IDEAPRENEURS | 50 COUNTRIES