

Boost your UI5 skills for creating accessibility apps

Dobrin Dimchev, SAP Labs Bulgaria
July 8th, 2022

Agenda

Accessibility Documentation

- Improvements in Demo Kit documentation
- End-user screen reader and keyboard handling documentation

Best Practice Application

- Accessibility-first samples

Accessibility Mission in SAP Discovery Center

- Learning accessibility in practice

Accessibility Documentation

- ▼ Essentials
 - ▼ Accessibility
 - › OpenUI5 Accessibility Features
 - Accessibility Support History
- ▼ Developing Apps
 - ▼ Accessibility
 - › Screen Reader Support
 - › Keyboard Handling
 - Colors and Theming
 - Text Size and Fonts
 - Recommendations
- ▼ Developing Controls
 - ▼ Accessibility Aspects
 - › Keyboard Handling for OpenUI5 Controls for Developers
 - › Screen Reader Support for OpenUI5 Controls
 - Theming

Accessibility Documentation

Understanding Accessibility UI5 Accessibility Building Blocks

The screenshot displays the SAP UI5 Accessibility Documentation page. The sidebar on the left contains a navigation menu with the following items: Model View Controller (MVC), Data Binding, Reusing UI Parts: Fragments, XML Templating, Working with Controls, Declarative Support, Error, Warning, and Info Messages, Routing and Navigation, Modules and Dependencies, Optimizing Applications, Adapting to Operating Systems And Devices, Testing, Theming, Localization, and Accessibility (which is currently selected). The main content area is titled 'Accessibility' and includes a link to 'Edit on GitHub'. The text explains that accessibility features are essential for the usability of each application and for users with disabilities. It mentions that OpenUI5 controls aim to comply with various accessibility standards such as screen reader support, high contrast theming, and keyboard handling. The page also features a section titled 'Understanding Accessibility' which defines accessibility as the possibility for everyone, including people who are differently-abled, to access and use information technology products. It states that accessibility is a guiding principle in OpenUI5 design and development processes and an integral part of the framework. The text further explains that accessibility is incorporated at two levels: framework and application level. Many features are available on the framework level and provided to app developers out of the box, which ensures consistency across all products. Some features need to be added or adapted according to the app-specific context and need to be considered when developing your app. A note mentions that depending on your use case, please refer to the corresponding accessibility chapter in the Developing Controls or Developing Apps sections in DemoKit. The page also includes a section titled 'OpenUI5 Accessibility Building Blocks' which lists several key features: Markup (correct HTML), ARIA attribute, Keyboard handling, Screen reader, Contrast, Resizing, and APIs. The page concludes with a 'Related information' section containing links to 'Accessibility in SAP Fiori #', 'Accessibility in the Developing Apps Section', and 'Accessibility Aspects in the Developing Controls Section'.

Accessibility

Accessibility features are essential for the usability of each application and essential for users with disabilities. In an ongoing approach, OpenUI5 controls aim to comply with various accessibility standards such as screen reader support, high contrast theming, and keyboard handling.

Understanding Accessibility

Accessibility refers to the possibility for everyone, including and especially people who are differently-abled, to access and use information technology products. It is a guiding principle in OpenUI5 design and development processes and integral part of the framework.

Accessibility is incorporated at two levels: framework and application level. Many features are available on the framework level and provided to app developers out of the box, which ensures consistency across all products. Some features need to be added or adapted according to the app-specific context and need to be considered when developing your app.

Depending on your use case please refer to the corresponding accessibility chapter in the Developing Controls or Developing Apps sections in DemoKit.

OpenUI5 Accessibility Building Blocks

- **Markup (correct HTML):** When developing accessible applications, you need to pay attention to the correctness of the resulting HTML. Some vital accessibility features (screen reader and keyboard support) rely on a correct and meaningful structure of the application.
- **ARIA attribute:** Accessible Rich Internet Applications (ARIA) is a set of attributes that define ways to make web content and applications more accessible to people with disabilities who use assistive technologies.
- **Keyboard handling:** Keyboard handling enables users to access every UI element of the application with the keyboard.
- **Screen reader:** A screen reader is a software application that identifies and interprets textual content on the UI. Developers need to know how the screen reader reads out the contents of the UI.
- **Contrast:** The different colors shown on the UI need to have a good contrast to each other, in order to be easily distinguishable.
- **Resizing:** Developers should always consider the fact that the application could be zoomed to 200%, and everything should still be visible.
- **APIs:** The framework provides APIs to help app developers improve the accessibility in the context of their application.

Related information

- [Accessibility in SAP Fiori #](#)
- [Accessibility in the Developing Apps Section](#)
- [Accessibility Aspects in the Developing Controls Section](#)

Accessibility Documentation

Accessibility Support History

- Enhancements
- Updates
- New accessibility concepts
- Important SAP Notes

 Documentation API Reference Samples Demo Apps Tools

Version 1.105.0 - development in progress. Change Version 🔍 📄 🔔 ⚙️

Accessibility Support History [Edit on GitHub](#) 🔄

OpenUI5 is following the SAP's design and development guidelines, as well as the testing procedures and accessibility reporting, that are based on WCAG 2.1 level A and AA.

The following table shows the availability of the different accessibility features.

Feature	Available as of version
HCW Theme	1.46
Screen Reader Support	1.30
Keyboard Handling	1.26
HCB Theme	1.26

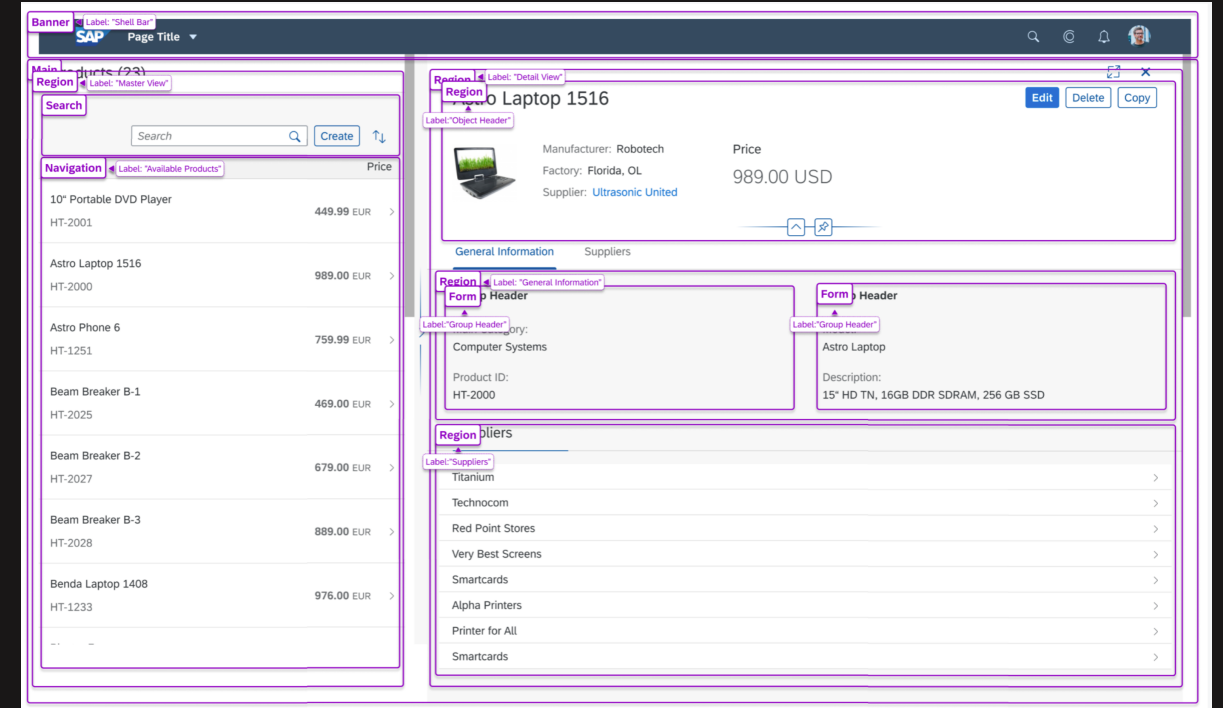
Enhancements and Updates	As of version
Update of the reference test environment. Latest update with UI5 1.102	For the versions, see SAP Note 2564165 .
We have prevented the overwrite of the Windows high contrast setting for all themes when using Chrome and Edge browsers.	1.101
We have enhanced OpenUI5 accessibility support according to the latest WAI-ARIA 1.1 specification	1.84
we have prevented the automatic insertion of role application on the body of the OpenUI5 applications. After the change, a mode-based screen reader will start operating in reading mode as its default mode. For more information, please refer to SAP Note 2925884 .	1.78
OpenUI5 is following the SAP's updated design and development guidelines, as well as the testing procedures and accessibility reporting, that are based on WCAG 2.1 level A and AA.	1.75

Accessibility Documentation

Developing Apps

- Logical groups
- New recommendations

- ▼ Developing Apps
 - ▼ Accessibility
 - ▼ Screen Reader Support
 - Landmark API
 - Labeling and Tooltips
 - Invisible Messaging
 - Dialogs, Popups, and Popovers
 - ▼ Keyboard Handling
 - Fast Navigation
 - Colors and Theming
 - Text Size and Fonts
 - Recommendations



Accessibility Best Practice Application

Accessibility Guide

The Accessibility Guide provides in-depth information with examples about web accessibility, whether you are a control or application developer, or a business user searching to learn more about web accessibility in the SAPUI5 area.

Accessibility Guide

Accessibility at Application Level

Many requirements are already covered by the [technology framework](#), but design aspects that are related to the individual purpose of the application still need to be implemented by the individual design teams. Here are some examples:

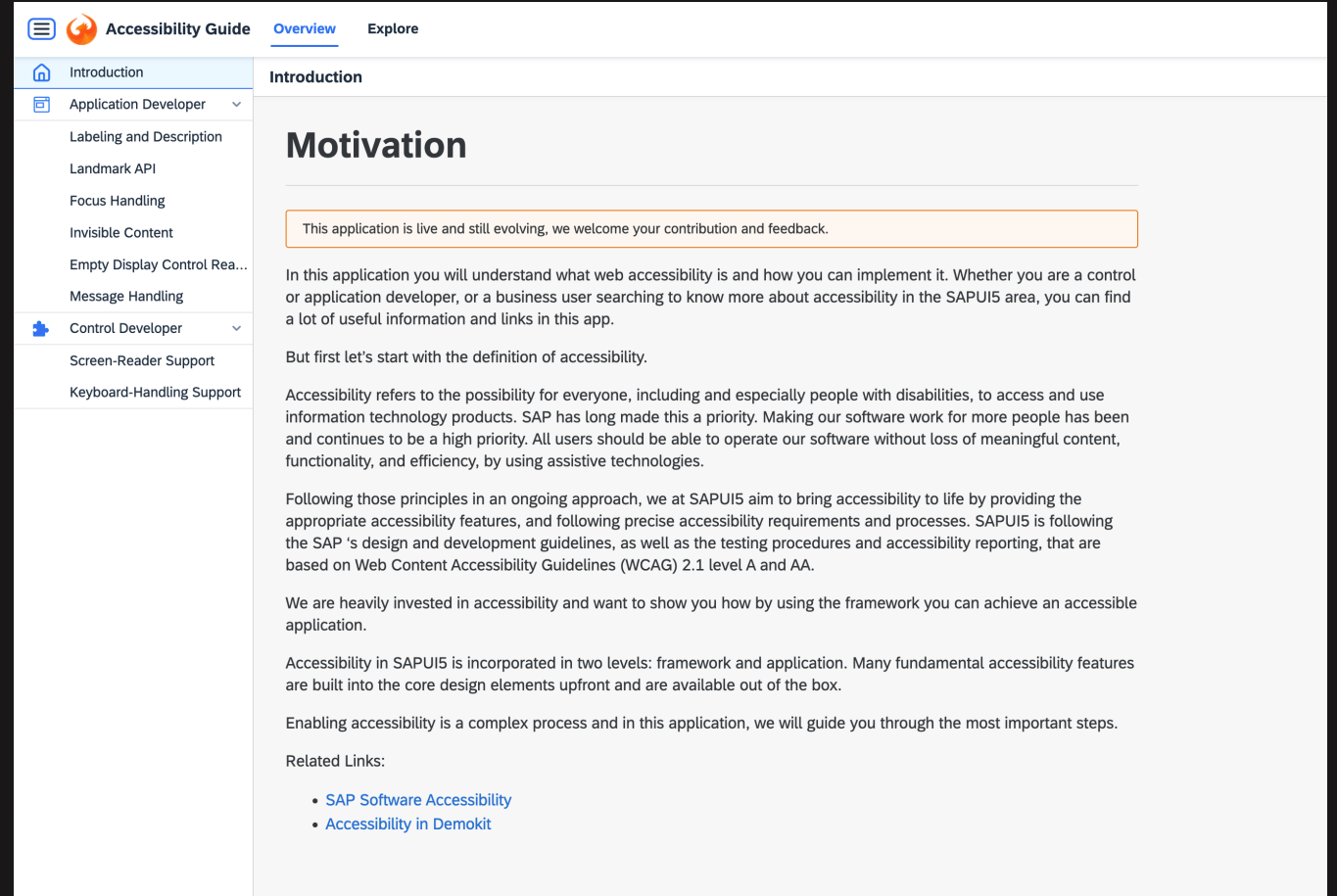
Keyboard Operation

- **Initial focus position:** To enable users to work through a task efficiently, always set an initial focus for the task. Set the focus to either the first logical interaction element or the first element in the task.
- **Structure and fast navigation:** Ensure that the navigation within your application is logical and follows the task structure for the intended purpose. When designing apps with a large set of functions or information blocks, always form logical groups to help users navigate efficiently. You can form logical groups using a container, a toolbar, or other grouping elements. To enable faster navigation, SAPUI5 allows users to skip groups (with F6 or SHIFT + F6). Also, enable direct keyboard navigation to logical groups, such as working areas or navigation areas.

Accessibility Best Practice Application

Located in Demo Kit Tools Page.

Provides **guidance** for developers to watch and follow best practices when developing accessibility features.



The screenshot displays the 'Accessibility Guide' application interface. On the left, a sidebar menu lists various topics: Introduction, Application Developer (selected), Labeling and Description, Landmark API, Focus Handling, Invisible Content, Empty Display Control Rea..., Message Handling, Control Developer, Screen-Reader Support, and Keyboard-Handling Support. The main content area is titled 'Introduction' and features a 'Motivation' section. This section includes a message box stating, 'This application is live and still evolving, we welcome your contribution and feedback.' Below this, the text explains that the application is designed to help users understand web accessibility and its implementation in the SAPUI5 environment. It emphasizes the importance of accessibility for all users and provides guidance on how to achieve an accessible application. The text also mentions that the application follows the SAP's design and development guidelines, as well as the WCAG 2.1 level A and AA standards. At the bottom, there are 'Related Links' pointing to 'SAP Software Accessibility' and 'Accessibility in Demokit'.

Accessibility Guide Overview Explore

Introduction

Motivation

This application is live and still evolving, we welcome your contribution and feedback.

In this application you will understand what web accessibility is and how you can implement it. Whether you are a control or application developer, or a business user searching to know more about accessibility in the SAPUI5 area, you can find a lot of useful information and links in this app.

But first let's start with the definition of accessibility.

Accessibility refers to the possibility for everyone, including and especially people with disabilities, to access and use information technology products. SAP has long made this a priority. Making our software work for more people has been and continues to be a high priority. All users should be able to operate our software without loss of meaningful content, functionality, and efficiency, by using assistive technologies.

Following those principles in an ongoing approach, we at SAPUI5 aim to bring accessibility to life by providing the appropriate accessibility features, and following precise accessibility requirements and processes. SAPUI5 is following the SAP 's design and development guidelines, as well as the testing procedures and accessibility reporting, that are based on Web Content Accessibility Guidelines (WCAG) 2.1 level A and AA.

We are heavily invested in accessibility and want to show you how by using the framework you can achieve an accessible application.

Accessibility in SAPUI5 is incorporated in two levels: framework and application. Many fundamental accessibility features are built into the core design elements upfront and are available out of the box.

Enabling accessibility is a complex process and in this application, we will guide you through the most important steps.

Related Links:

- [SAP Software Accessibility](#)
- [Accessibility in Demokit](#)

Accessibility Best Practice Application

Complete code examples

Semantically organized content

	View.view.xml	index.html	manifest.json	Component.js
Landmark API				
Dynamic Page				
Page				
Panel				
Object Page				
Labeling and Description				
Select				
Dialog				
Simple Form				
User Inputs				
Input with Description				

```
1 <mvc:View
2     xmlns="sap.m"
3     xmlns:mvc="sap.ui.core.mvc">
4 <App>
5 <Page
6     id="page"
7     backgroundDesign="Solid">
8 <landmarkInfo>
9 <PageAccessibleLandmarkInfo
10     rootRole="Region"
11     rootLabel="Product Details"
12     contentRole="Main"
13     contentLabel="Product Description"
14     headerRole="Region"
15     headerLabel="Product Header"
16     subHeaderRole="Region"
17     subHeaderLabel="Category Description"
18     footerRole="Region"
19     footerLabel="Product Footer"/>
20 </landmarkInfo>
```

	Controller.controller.js	View.view.xml	index.html
Landmark API			
Dynamic Page			
Page			
Panel			
Object Page			
Labeling and Description			
Select			
Dialog			
Simple Form			
User Inputs			
Input with Description			
Popover			
Popover with Description			
Icon-Only Buttons			
Mask Input			
Non-Decorative Images			
Focus Handling			
Popover Initial Focus Position			
Active Toolbar			
Keyboard Shortcuts			
Button with Shortcut			
Invisible Content			
Invisible Messaging			
Invisible Text			
Message Handling			
Message Handling Concept			
Message Toast			

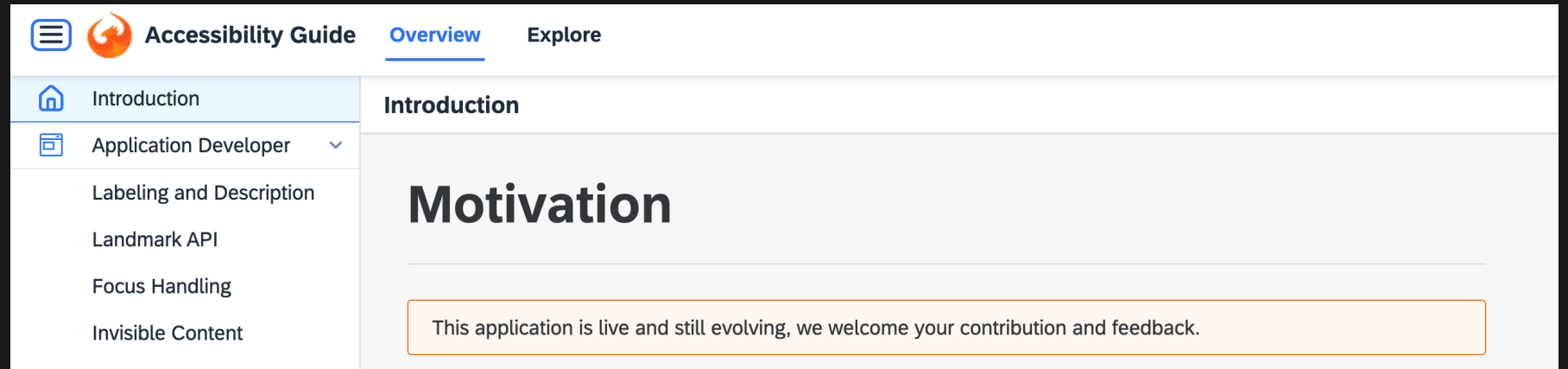
```
1 <mvc:View
2     height="100%"
3     controllerName="sap.m.sample.select.Controller"
4     xmlns:core="sap.ui.core"
5     xmlns:l="sap.ui.layout"
6     xmlns:mvc="sap.ui.core.mvc"
7     xmlns="sap.m"
8     class="sapUiContentPadding">
9 <core:InvisibleText text="Hello World" id="select-inv-text"></core:InvisibleText>
10
11 <l:VerticalLayout class="sapUiContentPadding">
12 <Label text="Using Label and labelFor property:" labelFor="select">
13 </Label>
14 <Select
15     id="select"
16     forceSelection="false"
17     selectedKey="{SelectedProduct}"
18     items="{
19         path: '/ProductCollection',
20         sorter: { path: 'Name' }
21     }">
22 <core:Item key="{ProductId}" text="{Name}" />
23 </Select>
24 </l:VerticalLayout>
25 <l:VerticalLayout class="sapUiContentPadding">
26 <Label text="Using InvisibleText and ariaLabelledBy:"></Label>
27 <Select
28     ariaLabelledBy="select-inv-text"
29     forceSelection="false"
30     selectedKey="{SelectedProduct1}"
31     items="{
32         path: '/ProductCollection1',
33         sorter: { path: 'Name' }
34     }">
35 <core:Item key="{ProductId}" text="{Name}" />
36 </Select>
37 </l:VerticalLayout>
38 </mvc:View>
```

Accessibility Best Practice Application

More to come

Live editor

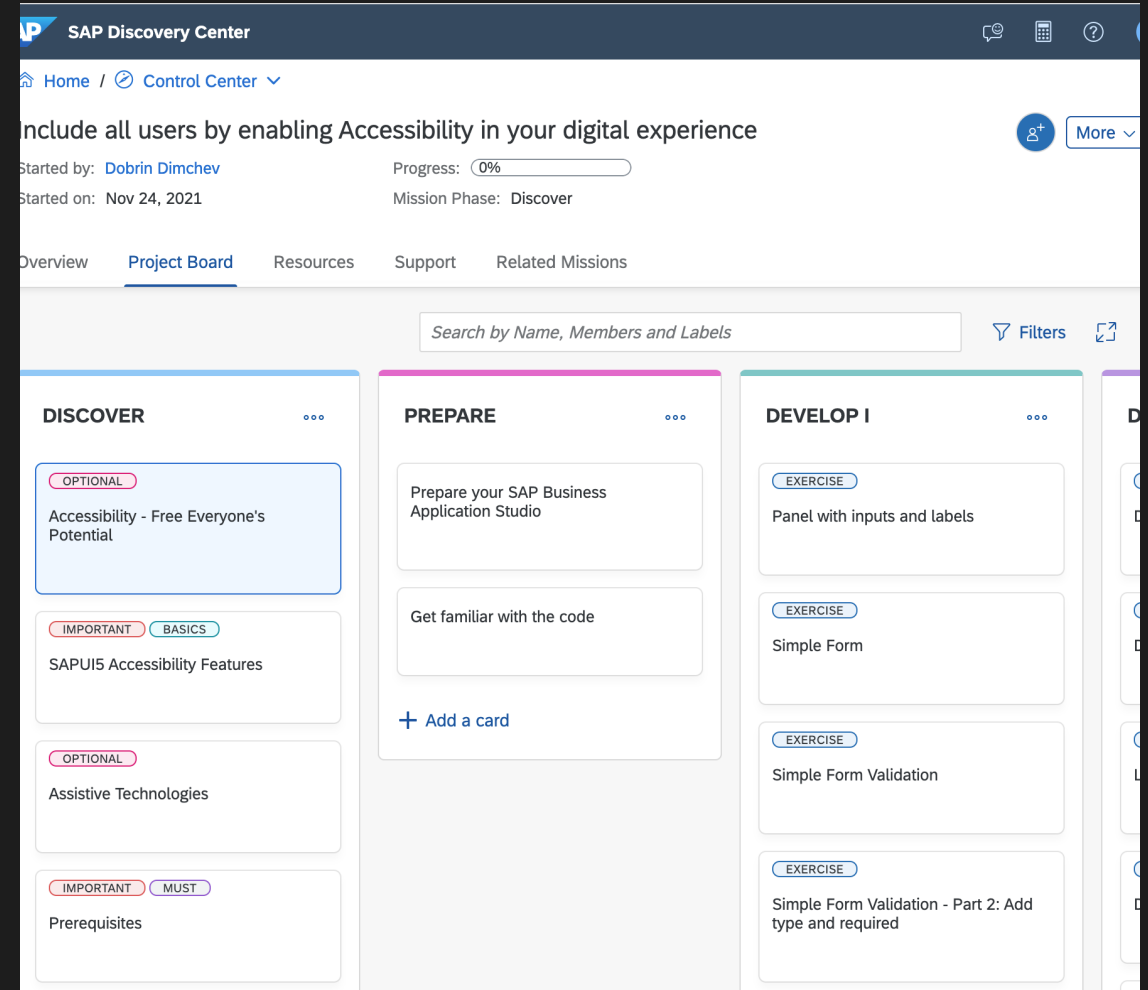
More contributions



Accessibility Mission in **SAP Discovery Center**

Accessibility mission in SAP Discovery Center

Bring more visibility about accessibility in UI5 and more easily and practically get to know the basic accessibility features and APIs that are offered by the framework.



Accessibility mission in SAP Discovery Center

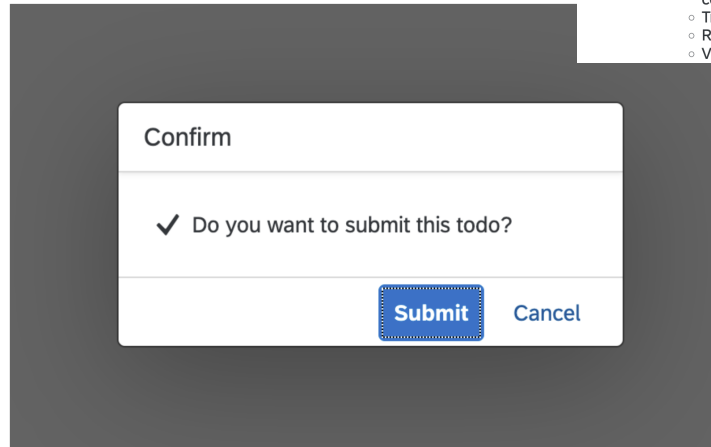
Develop

- Labeling
- Forms
- Validation
- Dialogs
- Messaging
- Landmark

Dialogs

Introduction

Dialogs are common way to provide information to the user after he interacts with That's why it is important to be fully accessible as very often the selections in them



Exercise

Panel with inputs and labels

Introduction

In the Todo application, users should first fill a simple form containing inputs and labels.

In this exercise, you are going to learn how to connect the Label controls with the Input controls manually. These examples are based on the Panel control and since it is a generic container the labelling functionality should be done manually. Later in this mission you will learn how to use the Form-based approach, and benefit from its features to automatically label the inputs.

Exercise

Initial setup

- Have the application served using the steps from the first exercise.
- Start Jaws screen reader.
- Explore the content of the Panel audible.

Connect the labels

- Problem: As you may have already noticed, in the initial test with Jaws, the labels of the fields are not read out. Your task is to connect the labels with the relevant input fields.
- Acceptance criteria: Jaws announces the labels of the input fields when they are focused.
- Hints:
 - Enhance the code in the *Todo.view.xml* file.
 - Read more about the Label control and check the available associations on this page: <https://ui5.sap.com/#/api/sap.m.Label%23associations>
 - Use the `labelFor` association. You should pass the ID of the labelled input. For the CheckBox you have two possibilities. The first one is to connect it with `labelFor` like it is described above. The other option is to use the `text` property of the CheckBox and the labelling will be done internally. You can read more about the properties here: <https://ui5.sap.com/#/api/sap.m.CheckBox%23controlProperties>.
 - Since `labelFor` accepts only one control, for the name inputs of the assignee, try to use the `ariaLabelledBy` association. Read more about the InvisibleText control: <https://ui5.sap.com/#/api/sap.ui.core.InvisibleText>.
 - Try to create two instances of the control with the relevant texts ("First Name" and "Last Name").
 - Reference the created controls via the `ariaLabelledBy` association of the inputs. You should pass the IDs of the newly created InvisibleTexts.
 - Validate the panel again with JAWS.

Key Takeaways & Resources

Accessibility Documentation

- <https://ui5.sap.com/#/topic/322f55d0cf1e4b459cc1911c899b7a5f>
- Essentials, Developing apps and controls
- Enhancements and updates
- Refers to best practice samples

Accessibility Best Practice Application

- <https://ui5.sap.com/test-resources/sap/m/demokit/accessibilityGuide/webapp/index.html>
- Complete accessibility-first samples

Accessibility Mission

- <https://discovery-center.cloud.sap/missiondetail/3530/3571/>
- Make your application more accessible to all users

Thank you!

Dobrin Dimchev

JavaScript Developer

dobrin.dimchev@sap.com