

openSAP Evolved Web Apps with SAPUI5

Week 3 Unit 5: Scaling Up with UI Reuse patterns

Exercises

PUBLIC





TABLE OF CONTENTS

SCALING UP WITH UI5 REUSE PATTERNS.....	5
Create an order preparation control composed from existing UI5 controls	6
Add the order preparations control to the application.....	8
RELATED MATERIAL.....	10





SCALING UP WITH UI5 REUSE PATTERNS

Summary

The application for UI5 Air Cargo already has a great feature set. To improve this even further, we will introduce a new UI control in this unit, which gives the users an overview about the current state of order preparation. They need to know if the items are packed, the invoice is printed and if a product sample has been added.

To fulfill those requirements, we will implement an XML composite control which consists of several existing UI5 controls. We will then incorporate the new control into the application.

Preview

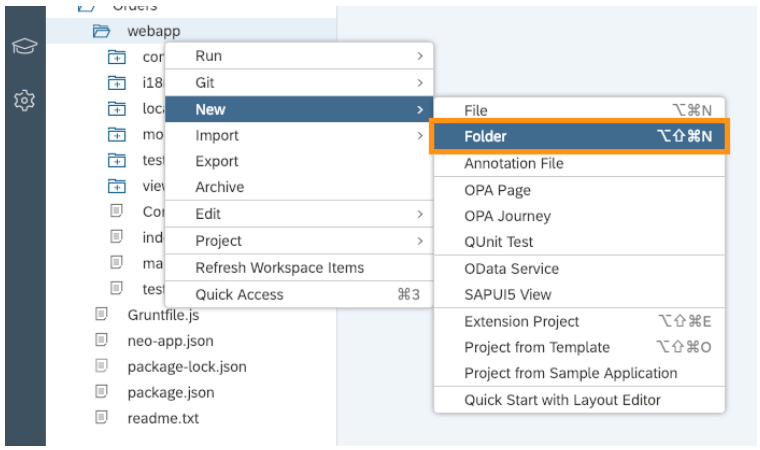
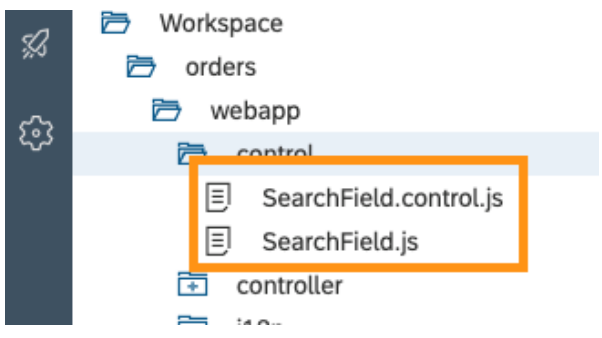
The order preparation control will be composed of layouts, switches and a button and will be placed above the table in the detail part of the application.

The screenshot shows a web application interface for 'PicoBit'. At the top, there's a header with the title 'PicoBit' and a 'Delivery status:' label with a blue 'NEW' button. Below this is a section titled 'Order Preparations' with a dropdown arrow. Inside this section, there are three steps: '1. Pack Items' with a green checkmark and a green circle, '2. Print Invoice' with a green checkmark and a green circle, and '3. Add Sample' with a red 'X' and a red circle. To the right of these steps is a blue 'Confirm' button. Below the 'Order Preparations' section is a table titled 'Order Items (4)'. The table has four columns: 'Quantity', 'ProductId', 'Delivery date', and 'Delivery urgency'.

Quantity	ProductId	Delivery date	Delivery urgency
----------	-----------	---------------	------------------

Order preparation control as `sap.ui.core.XMLComposite`

Create an order preparation control composed from existing UI5 controls

Explanation	Screenshot
<p>1. Go to the webapp folder and create a new folder called control. This is where our new control will be located. For custom controls in your application it is a best practice to always put them in a dedicated folder.</p>	
<p>2. In the new folder, add two new files:</p> <ul style="list-style-type: none"> OrderPreparations.js OrderPreparations.control.xml 	

webapp/control/OrderPreparations.js (NEW)

```
sap.ui.define([
    "sap/ui/core/XMLComposite"
], function(XMLComposite){
    "use strict";

    return XMLComposite.extend("opensap.orders.control.OrderPreparations", {
        metadata: {
            properties: {
                switchStateItems: { type:"boolean", defaultValue: false},
                switchStateInvoice: { type:"boolean",defaultValue: false},
                switchStateSample: { type:"boolean", defaultValue: false}
            },
            events: {
                confirm: {}
            }
        },
        onConfirm: function() {
            this.fireEvent("confirm", {});
            this.byId("orderPreparations").setExpanded(false);
            this._resetSwitches();
        }
    });
});
```

```

        reset: function() {
            this.byId("orderPreparations").setExpanded(true);
            this._resetSwitches();
        },

        _resetSwitches: function() {
            this.setSwitchStateItems(false);
            this.setSwitchStateInvoice(false);
            this.setSwitchStateSample(false);
        }

    });
});

```

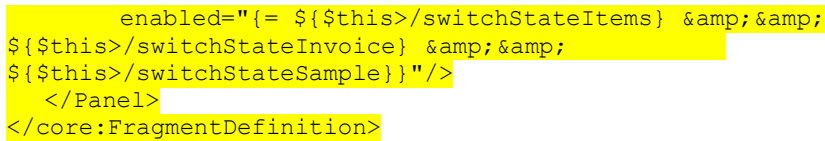
In the `sap.ui.core.XMLComposite` control implementation you need to define the properties like in any other control. These are used in the `control.xml` file, which contains the declarative description of the control in the well-known XML markup.

webapp/control/OrderPreparations.control.xml (NEW)

```

<core:FragmentDefinition
    xmlns="sap.m"
    xmlns:core="sap.ui.core"
    xmlns:l="sap.ui.layout">
    <Panel
        headerText="{i18n>OrderPreparationTitle}"
        expandable="true"
        expanded="true"
        id="orderPreparations">
        <l:HorizontalLayout allowWrapping="true">
            <Label text="{i18n>PackItemsTitle}"
                class="sapUiSmallMargin"/>
            <Switch
                id="packItemsSwitch"
                state="{ $this>/switchStateItems}"
                type="AcceptReject"/>
        </l:HorizontalLayout>
        <l:HorizontalLayout allowWrapping="true">
            <Label text="{i18n>PrintInvoiceTitle}"
                class="sapUiSmallMargin"/>
            <Switch
                id="printInvoiceSwitch"
                state="{ $this>/switchStateInvoice}"
                type="AcceptReject"/>
        </l:HorizontalLayout>
        <l:HorizontalLayout allowWrapping="true">
            <Label text="{i18n>AddSampleTitle}"
                class="sapUiSmallMargin"/>
            <Switch
                id="addSampleSwitch"
                state="{ $this>/switchStateSample}"
                type="AcceptReject"/>
        </l:HorizontalLayout>
        <Button
            id="confirm"
            text="{i18n>ConfirmPreparations}"
            type="Emphasized"
            class="sapUiSmallMarginBegin"
            press=".onConfirm"

```



Note: The `$this` is the reference to a `sap.ui.model.base.ManagedObjectModel` which is the underlying connection between the `OrderPreparations.js` implementation and the declarative `Orderpreparations.control.xml`. It provides you easy access to its properties and aggregations through data binding. Take a look at how the properties are bound to the `$this` model.

To be able to show our new control in the application appropriately positioned, add a layout and make sure, that it is displayed above the existing table in the detail view. Also add an event handler to the detail controller, to be able to show a message toast as soon as the order preparation has been completed. Finally add also the according translations in the i18n properties file.

```
<mvc:View
    controllerName="opensap.orders.controller.Detail"
    xmlns="sap.m"
    xmlns:semantic="sap.f.semantic"
    xmlns:tnt="sap.tnt"
    xmlns:si="sap.suite.ui.commons.statusindicator"
    xmlns:mvc="sap.ui.core.mvc"
    xmlns:l="sap.ui.layout"
    xmlns:control="opensap.orders.control">

<semantic:SemanticPage
    id="detailPage"
    busy="{detailView>/busy}"
    busyIndicatorDelay="{detailView>/delay}"
    showFooter="true">
    <semantic:titleHeading>
        <Title text="{CustomerName}"/>
    </semantic:titleHeading>
    <semantic:headerContent>
        <ObjectAttribute title="{i18n>StatusDesc}"/>
        <tnt:InfoLabel
            text="{LifecycleStatusDescription}"
            colorScheme="{
                path: 'LifecycleStatusDescription',
                formatter: '.formatter.deliveryStatus'
            }"/>
    </semantic:headerContent>
    <semantic:content>
        <l:VerticalLayout>
            <control:OrderPreparations
                id="orderPreparations"
                confirm=".onConfirm">
            </control:OrderPreparations>
            <Table
```



```

    </Table>
  </l:VerticalLayout>
</semantic:content>
...

```

webapp/view/Detail.controller.js

```

sap.ui.define([
  "./Basecontroller",
  "sap/ui/model/json/JSONModel",
  "../model/formatter",
  "sap/m/library",
  "sap/m/Device",
  "sap/m/MessageToast"
], function (Basecontroller, JSONModel, formatter, mobileLibrary, Device,
  MessageToast) {
  "use strict";

  return Basecontroller.extend("opensap.orders.controller.Detail", {

    formatter: formatter,

    /* ===== */
    /* lifecycle methods */
    /* ===== */

    ...

    onConfirm: function (oEvent) {
      var oBinding = oEvent.getSource().getBindingContext().getObject();
      var oMessage = this.getResourceBundle().getText(
        "OrderPreparationMessage", [oBinding.CustomerID,
          oBinding.CustomerName]);
      MessageToast.show(oMessage);
    }

  });
});

```

webapp/i18n/i18n.properties

```

# This is the resource bundle for Browse Orders

#XTIT: Application name
appTitle=Browse Orders

...

#~~~ OrderPreparation control ~~~~~

#XTIT: Title text for the Order Preparation process
OrderPreparationTitle=Order Preparations

#XTIT: Confirm Order Preparation Message
OrderPreparationMessage=Order {0} for {1} is prepared for shipment!

```

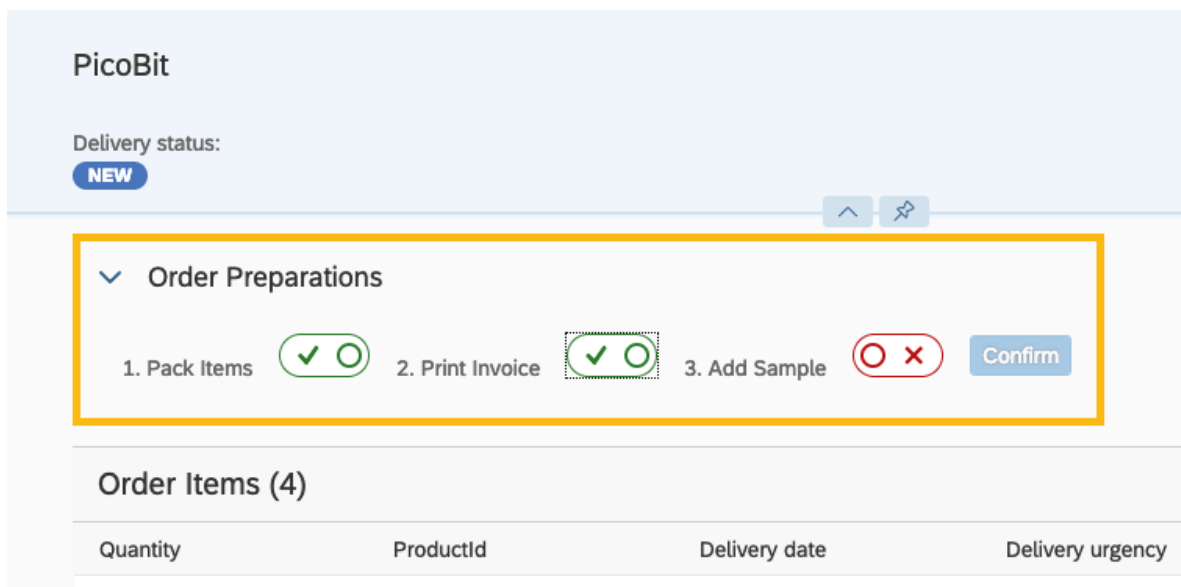
```
#XTIT: Title text for the Pack Items Switch
PackItemsTitle=1. Pack Items

#XTIT: Title text for the Print Invoice Switch
PrintInvoiceTitle=2. Print Invoice

#XTIT: Title text for the Add Sample Switch
AddSampleTitle=3. Add Sample

#XBUT: Text for the Confirm order preparations button
ConfirmPreparations=Confirm
```

Now you are ready to see the `OrderPreparations` control in action. Run the application and have a look at the results. If it looks like in the preview screenshot, you did everything right. Congratulations!



Order preparation control as `sap.ui.core.XMLComposite`

RELATED MATERIAL

- [Demo Kit: Standard Composite Controls](#)
- [Demo Kit: XML Composite controls](#)
- [Demo Kit: The library.js File](#)
- [Documentation: Community Project UI5Lab](#)
- [GitHub: UI5Lab Library Example](#)
- [GitHub: Control Libraries Documentation](#)

Coding Samples

Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a production system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

www.sap.com/contactsap

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.