This chapter walks you through developing a plug-in and then deploying it on SAP Cloud Platform or SAP NetWeaver AS ABAP. It also provides a quick look at the predefined plug-ins provided by SAP.
Chapter 7
Plug-Ins

Plug-ins help you run custom code just after SAP Fiori Launchpad has been initialized. In this chapter, you’ll learn how to develop a plug-in and deploy it on both SAP Cloud Platform and SAP NetWeaver AS ABAP.

Plug-ins are SAPUI5 components without a user interface. They use a special intent, such as shell plugin, when deploying on SAP Fiori launchpad. All plug-ins will be loaded, and the initialize code in those plug-ins will be executed after SAP Fiori launchpad has been loaded.

A plug-in is a major container for code that changes SAP Fiori launchpad globally. Plug-ins have the following characteristics:

- Plug-ins are automatically loaded and initialized when SAP Fiori launchpad is started.
- Plug-ins are implemented as SAPUI5 components and provide all standard deployment and lifecycle features of SAPUI5.
- Plug-ins will always be implemented in a platform-independent way, but platform-specific configuration is allowed. On SAP Cloud Platform, you need to configure plug-ins on SAP Cloud Platform Portal. On SAP NetWeaver AS ABAP, you configure them via SAP Fiori launchpad designer.
- On SAP NetWeaver AS ABAP, plug-ins can be enabled and configured dynamically by assigning users to roles.

In this chapter, we’ll walk you through developing a plug-in and then deploying it on SAP Cloud Platform or SAP NetWeaver AS ABAP. We’ll end the chapter with a quick look at the predefined plug-ins provided by SAP.

7.1 Developing a Plug-In

This section will walk you through developing a plug-in, from using templates and adjusting your code through testing the plug-in to ensure it’s working properly.
7.1.1 Creating a Plug-In Using a Template

The easiest way to create a new plug-in is by using the template provided by SAP Web IDE full-stack version. Because it's normally not recommended to write complex code in a plug-in, you can always create plug-ins based on the template.

The template not only contains the basic structure and mandatory code for initializing the component but also can help you generate sample code to perform the following activities:

- Adding a button to the SAP Fiori launchpad header
- Adding an SAP Fiori launchpad footer with a button
- Adding buttons to the Me Area

The following procedure will guide you through the basic process of creating a plug-in:

1. Enter SAP Web IDE full-stack version.
2. Follow menu path File • New • Project from Template, as shown in Figure 7.1.

   ![Figure 7.1 Create Project from Template](image1)

3. In the Template Selection window, select SAP Fiori Launchpad Plug-In and click Next, as shown in Figure 7.2.

4. In the Basic Information window, enter "FlpPlugin" for the Project Name and then click Next, as shown in Figure 7.3.

![Figure 7.2 Template Selection](image2)

![Figure 7.3 Basic Information](image3)
5. In the **Template Customization** window, enter “flpdev.flpplugin” for **Plugin ID** and any text of your choice for **Title**. Here you can choose the **Sample Code** you need. For this example, select **Add Button to Launchpad Header**, then click **Finish**, as shown in Figure 7.4.

![Template Customization](image)

**Figure 7.4** Template Customization

Now that you’ve created your first plug-in, let’s look at the generated code (see Listing 7.1). The most important file for the plugin is `component.js`

```javascript
// This method help you get a deferred object which will return the renderer object

.sap.ui.define([
  "sap/ui/core/Component",
  "sap/m/Button",
  "sap/m/Bar",
  "sap/m/MessageToast"
], function (Component, Button, Bar, MessageToast) {

  return Component.extend("flpdev.flpplugin.Component", {

    metadata: {
      "manifest": "json"
    },

    init: function () {
      var rendererPromise = this._getRenderer();

      // This is example code. Please replace with your implementation!

      /**
      * Add item to the header
      */
      rendererPromise.then(function (oRenderer) {
        // Here is the place for your custom code
        oRenderer.addHeaderItem({
          icon: "sap-icon://add",
          tooltip: "Add bookmark",
          press: function () {
            MessageToast.show("This SAP Fiori launchpad has been extended to improve your experience");
          }
        }, true, true);
      });

      },

      /**
      * This method help you get a deferred object which will return the renderer object
      */

      _getRenderer: function () {
        // You can just ignore the generated code, just use it!
        return rendererPromise;
      }
    });

    });
```

Listing 7.1 FlpPlugin: Generated Component

Now that you’ve created your first plug-in, let’s look at the generated code (see Listing 7.1). The most important file for the plugin is `component.js`.

```javascript
sap.ui.define([
  "sap/ui/core/Component",
  "sap/m/Button",
  "sap/m/Bar",
  "sap/m/MessageToast"
], function (Component, Button, Bar, MessageToast) {

  return Component.extend("flpdev.flpplugin.Component", {

    metadata: {
      "manifest": "json"
    },

    init: function () {
      var rendererPromise = this._getRenderer();

      // This is example code. Please replace with your implementation!

      /**
      * Add item to the header
      */
      rendererPromise.then(function (oRenderer) {
        // Here is the place for your custom code
        oRenderer.addHeaderItem({
          icon: "sap-icon://add",
          tooltip: "Add bookmark",
          press: function () {
            MessageToast.show("This SAP Fiori launchpad has been extended to improve your experience");
          }
        }, true, true);
      });

      },

      /**
      * This method help you get a deferred object which will return the renderer object
      */

      _getRenderer: function () {
        // You can just ignore the generated code, just use it!
        return rendererPromise;
      }
    });

    });
```

Listing 7.1 FlpPlugin: Generated Component
Note the following elements of this file:

- The component is a subclass of `sap.ui.core.Component`. In contrast, other SAPUI5 applications are subclasses of `sap.ui.core.UIComponent`. A plug-in will never have a view or controllers; it inherits from `component` so that the request payload can be minimized.

- A `_getRenderer` method is generated for you. The most common usage for a plug-in is to extend SAP Fiori launchpad, so the renderer is the most used object. Unlike normal applications, a plug-in is executed during the startup of SAP Fiori launchpad, so it’s not certain that the renderer object has been loaded when the plug-in is executing. The `_getRenderer` method returns a deferred object that helps you make sure your code is executed after the renderer has been loaded.

- In the `init` method, some example code has been generated; all your code that accesses the SAP Fiori launchpad should be placed in the functions after resolving the `rendererPromise` object.

### 7.1.2 Adjusting Implementation Code

Now let’s focus on the core of the code (see Listing 7.2), in which you can find in `init` method adds a header item.

```javascript
oRenderer.addHeaderItem({
    icon: "sap-icon://add",
    tooltip: "Add bookmark",
    press: function () {
        MessageToast.show("This SAP Fiori launchpad has been extended to improve your experience");
    }
}, true, true);
```

Listing 7.2 FlpPlugin: Code Generated for Adding Header Item

As you’ve already seen, it’s not easy to test this because the header item will not display if there isn’t enough space. Let’s change the code where you found in `init` method to add a header end item, as shown in Listing 7.3.

```javascript
oRenderer.addHeaderEndItem("sap.ushell.ui.shell.ShellHeadItem", {
    id: "addbookmarkitem",
    icon: "sap-icon://add",
    tooltip: "Add bookmark",
    press: function () {
        MessageToast.show("This SAP Fiori launchpad has been extended to improve your experience");
    }
}, true, true);
```

Listing 7.3 FlpPlugin: Add Header End Item

### 7.1.3 Testing Your Plug-In

Testing of a plug-in is a little different compared to testing a normal SAPUI5 application. The following procedure will guide you through the testing process:

1. Right-click `Component.js` and choose `Run • Run Configurations` in the context menu, as shown in Figure 7.5.

   ![Figure 7.5 Run Configuration](image)

2. Click the Plus button and choose `Run as SAP Fiori Launchpad Sandbox` in the pop-over, as shown in Figure 7.6.
3. For the **File Name** field, choose `/fioriSandboxConfig.json` from the dropdown list. Keep the other fields unchanged, as shown in Figure 7.7.

4. Switch to the **URL Components** tab and set the value of the **URL Hash Fragment** field to `#Shell-home`, as shown in Figure 7.8.

5. Click **Save and Run**. 

The reason for a test like this is that the plug-in doesn’t have a UI. The `fioriSandboxConfig.json` file is a configuration file that loads the plug-in. To make sure the index page displays, you need to specify the intent as `#Shell-home` in the run configuration.

### 7.2 Deploying the Plug-In on SAP Cloud Platform

After development and testing, it’s time to deploy your plug-in to SAP Fiori launchpad. First, you need to deploy your plug-in as an SAPUI5 application on SAP Cloud Platform, then you need to add an app as a plug-in in SAP Fiori configuration cockpit for your SAP Fiori launchpad site.
7.2.1 Deployment and Activation

As we mentioned in earlier chapters, you need to deploy your plug-in as an SAPUI5 application to SAP Cloud Platform. Proceed as follows:

1. In your SAP Web IDE full-stack version, keep your project for the plug-in selected. Choose menu path Deploy > Deploy to SAP Cloud Platform, as shown in Figure 7.9.

   ![Figure 7.9 Deploy to SAP Cloud Platform](image)

2. On the Deploy Application to SAP Cloud Platform page, keep everything unchanged and click Deploy, as shown in Figure 7.10.

   ![Figure 7.10 Deploy Application to SAP Cloud Platform: Application Details](image)

3. Once you receive the message that your application has deployed successfully, click Close, as shown in Figure 7.11.

   ![Figure 7.11 SAP Fiori Application Successfully Deployed](image)

4. Enter the management page of your SAP Fiori launchpad site and follow menu path Content Management > Apps.

5. Click the Plus button to add an app and open the search help for App Resource, as shown in Figure 7.12.

   ![Figure 7.12 Add New App](image)

Note

The plug-in is not an SAPUI5 application, so it can’t be registered to SAP Fiori launchpad here. You must do it manually.
6. In the value help dialog, search for “flp plugin”, then choose the line that represents your plug-in and click **OK**, as shown in Figure 7.13.

![Figure 7.13 Select Your Plug-In](image)

7. Scroll down to the **App Resources Details** section. For **App Type**, choose **Shell Plugin** from the dropdown list. For **Shell Plugin Type**, choose **Custom**. Leave the other fields unchanged, as shown in Figure 7.14.

![Figure 7.14 Set App as Plug-In](image)

8. Switch to the **Catalogs** tab and click the **Plus** button to assign the app to a tile catalog, as shown in Figure 7.15.

![Figure 7.15 Assign Catalog](image)

9. Select **Sample Catalog** and click **OK** in the **Select Catalogs** pop-up shown in Figure 7.16.

![Figure 7.16 Select Catalogs](image)
10. Save all your changes by clicking **Save** on the footer toolbar.
11. Click the globe button in the top-right corner to open the **Publish Site** pop-up window.
12. Check **Clear HTML5 Application Cache**, then click **Publish and Open**, as shown in Figure 7.17.

![Publish Site](image)

**Figure 7.17** Publishing Site

13. The result should look like Figure 7.18.

![Plug-In in SAP Cloud Platform Portal](image)

**Figure 7.18** Plug-In in SAP Cloud Platform Portal

### 7.2.2 Avoiding Multiple Code Executions

As you can see, the plug-in is easy to configure. But there are potential risks to configuring a plug-in more than once. The following example will configure and deploy the app again, with some unintended consequences:

1. Return to your configuration page for your SAP Fiori launchpad site.
2. Add an app again like in the previous demo. The only difference is that you need to change **App Title** to “flpplugin2”, as shown in Figure 7.19.

![App Resource Details](image)

**Figure 7.19** Configure Another App for Same Plug-In

3. Clear the cache and publish your site again; you’ll see two items appearing at the end of the shell header, as shown in Figure 7.20.

![Result of Configuring Plug-Ins Twice](image)

**Figure 7.20** Result of Configuring Plug-Ins Twice
To solve this problem, you need to ensure your code executes only once regardless of how many times the plug-in is executed. A general approach is to check for the existence of the control you want to add before adding it. To do so, switch back to SAP Web IDE full-stack version, then change the `init` method of `component.js` as shown in Listing 7.4.

```javascript
init: function () {
    var rendererPromise = this._getRenderer();
    rendererPromise.then(function (oRenderer) {
        // get reference of the item
        var oItem = sap.ui.getCore().byId("addbookmarkitem");
        // Add item only if the item does not exist
        if (!oItem) {
            oRenderer.addHeaderEndItem("sap.ushell.ui.shell.ShellHeadItem",
                {
                    id: "addbookmarkitem",
                    icon: "sap-icon://add",
                    tooltip: "Add bookmark",
                    press: function () {
                        MessageToast.show("This SAP Fiori launchpad has been extended to improve your experience");
                    },
                    true, true);
        }
    });
},
Listing 7.4 FlpPlugin: Code for Avoiding Multiple Executions

Deploy your plug-in again and publish your site once more. You’ll find that the issue has been solved.

### 7.2.3 Working with Configurable Parameters

By working with configurable parameters, you can make your plug-in more flexible. In this section, you’ll modify your plug-in to work with configurable parameters, as follows:

1. Return to your SAP Fiori launchpad site management page. Follow menu path `Content Management • Apps`, then select `flpplugin` in the app list. Enter edit mode by clicking `Edit`.

2. Switch to the `Parameters` tab. Click the `Plus` button on the toolbar of the `Intent Parameters` table. Enter “icon” in the `Name` column and “accept” in the `Default Value` column. Then click `Save` to save all changes. The form should look like Figure 7.21.

![Figure 7.21 Set Parameter for Plug-In](image)

3. To avoid confusion, delete the `flpplugin2` app by clicking on it in the apps list and clicking `Delete`, as shown in Figure 7.22.

![Figure 7.22 Delete flpplugin2](image)
4. Switch to the SAP Web IDE and change the code of the `init` method of `component.js` as shown in Listing 7.5. Here we get parameter `icon` as the icon's name, after which we can display the icon according to user settings.

```javascript
init: function () {
  var rendererPromise = this._getRenderer();
  // Get configurable parameter and generate the icon URL
  var oComponentData = this.getComponentData();
  var icon = oComponentData.config.icon;
  var iconUrl = "sap-icon://" + icon;
  rendererPromise.then(function (oRenderer) {
    var oItem = sap.ui.getCore().byId("addbookmarkitem");
    if (!oItem) {
      oRenderer.addHeaderEndItem("sap.ushell.ui.shell.ShellHeadItem", {
        id: "addbookmarkitem",
        // Change the static parameter to variable
        icon: iconUrl,
        tooltip: "Add bookmark",
        press: function () {
          MessageToast.show("This SAP Fiori launchpad has been extended to improve your experience");
        },
      });
    }
  });
},
```

Listing 7.5 FlpPlugin: Read Parameters

5. Deploy your plug-in again to SAP Cloud Platform and republish your site, as shown in Figure 7.23.

6. As a result, the icon for the item you’ve added will change according to your configuration.

### 7.3 Deploying the Plug-In on SAP NetWeaver AS ABAP

A change in code level isn’t needed to deploy your plug-in on SAP NetWeaver AS ABAP. The only difference is the method of deployment and configuration. In this section, we’ll guide you through these processes.

#### 7.3.1 Deployment

To deploy your plug-in to SAP NetWeaver AS ABAP, you first need to deploy it to the ABAP Repository as a BSP (Business Server Page) application. Then use SAP Fiori launchpad designer to create a target mapping using a predefined intent called shell plugin. To begin the deployment, follow these steps:

1. Switch back to SAP Web IDE full-stack version and choose **Deploy > Deploy to SAPUI5 ABAP Repository**, as shown in Figure 7.24.

![Menu for Deploying to ABAP](image)

2. In the **Deployment Options** step, choose your system and select **Deploy a New Application**. Then click **Next**, as shown in Figure 7.25.
3. In the **Deploy a New Application** step, enter “zplugin” for **Name** and text of your choice for **Description**. Select **$TMP** by clicking the **Browse** button to the right of the **Package** field. Then click **Next**, as shown in Figure 7.26.

4. Finally, click **Finish** in the **Confirm** step, as shown in Figure 7.27. Your application has now been successfully deployed!

---

### 7.3 Deploying the Plug-In on SAP NetWeaver AS ABAP

#### 7.3.2 Configuration

With the plug-in deployed, you can start to configure and activate the plug-in for your user. A predefined intent is reserved for plug-ins: shell plugin. To begin, follow these steps:

1. Open SAP Fiori launchpad designer on your local computer.
2. Add a tile catalog by clicking the **Add** button in the footer toolbar of the catalog list. Use “MyPlugin” for the **Title** and “ZPLUGIN” for the **ID**. Then click **Save** in the pop-up window, as shown in Figure 7.28.
3. Switch to the Target Mapping tab and click Create Target Mapping in the footer toolbar, as shown in Figure 7.29.

4. Fill in the form using the information in Table 7.1.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic Object</td>
<td>Shell</td>
</tr>
<tr>
<td>Action</td>
<td>Plugin</td>
</tr>
<tr>
<td>Application Type</td>
<td>SAPUI5 Fiori App</td>
</tr>
<tr>
<td>Title</td>
<td>AddHeaderEndItem</td>
</tr>
<tr>
<td>URL</td>
<td>/sap/bc/ui5_ui5/sap/zplugin</td>
</tr>
<tr>
<td>ID</td>
<td>flpdev.flpplugin</td>
</tr>
</tbody>
</table>

Table 7.1 Properties for Plug-In Target Mapping

5. In the Parameters table, add a parameter, entering “icon” for Name and “accept” for Default Value. The result should look like Figure 7.30.

6. Save, and close SAP Fiori launchpad designer.
7. Logon to the SAP GUI and enter Transaction PFCG.
8. Enter “ZROLE_PLUGIN” for the Role name and click the Single Role button, as shown in Figure 7.31.

![Figure 7.31 Create New Role](image1)

9. Provide a Description and click Save, as shown in Figure 7.32.

![Figure 7.32 Save Role before Continuing](image2)

10. Switch to the Menu tab, click the small triangle after button Transaction, and select SAP Fiori Tile Catalog, as shown in Figure 7.33.

![Figure 7.33 Add Tile Catalog as Menu](image3)

11. In the pop-up window, enter “ZPLUGIN” for Catalog ID, then click the green checkmark to confirm, as shown in Figure 7.34.

![Figure 7.34 Select Catalog](image4)
12. The result should look like Figure 7.35.

![Figure 7.35 Result of Adding Tile Catalog](image)

13. Switch to the **User** tab, add your user to the list, and save the role, as shown in Figure 7.36.

![Figure 7.36 Add User and Save](image)

14. Switch back to your browser and enter SAP Fiori launchpad on your SAP NetWeaver AS ABAP system. The SAP Fiori launchpad will now look like Figure 7.37.

![Figure 7.37 Result of Plug-In Activated on SAP NetWeaver AS ABAP](image)

### 7.4 Predefined Plug-Ins

There are predefined plug-ins for special purposes in the system. For example, a plug-in for setting user defaults is available for SAP S/4HANA Finance to help users set default values for common parameters. Another plug-in is for enabling or disabling runtime authoring tools for a specific user.

#### 7.4.1 Plug-In for Setting User Defaults

This plug-in adds setting options in the user setting dialog. It consists of common finance parameters like company code.

If you’re using SAP Fiori launchpad on SAP NetWeaver AS ABAP and all apps for SAP S/4HANA have been installed, you can find the plug-in in business catalog `SAP_SFIN_BC_USER_PARAM`.

For SAP Fiori launchpad on SAP Cloud Platform, you need to perform the following steps:

1. Import the Default Values shell plug-in app using the Transport Manager.

**Note**

This feature is only available if you’re subscribed to SAP Fiori Cloud for SAP S/4HANA Finance content.
2. In the Configure Apps editor, select the **Parameters** tab. For each application that was developed to support user defaults as a navigation parameter, populate the parameter fields using the following structures:

- **Parameter name:** `<parametername>`
- **Parameter default value:** `userdefault.<parametername>`

These parameters enable the apps to use the Default Values shell plug-in.

### 7.4.2 Plug-In for Activating Runtime Authoring

Flexibility is key! Enterprise software must adapt to rapidly changing environments. For example, customers need their apps to fit their processes without long IT projects to adapt them, and cloud providers want to run the same software for everyone to reduce total cost of ownership (TCO). Do you think adapting the user interface of SAP Fiori apps (e.g., by adding, hiding or rearranging fields) is a complex process? Think again! SAPUI5 flexibility services allow for upgrade-safe and modification-free UI changes on different levels (e.g., on the customer side) that can be performed by different users (end users, key users, and developers).

The plug-in for runtime authoring can enable or disable key users from creating their own adaptations of an SAP Fiori application based on the SAPUI5 flexibility service.

In SAP NetWeaver AS ABAP, there’s already a tile catalog called `SAP_UI_FLEX_KEY_USER`. A plug-in is contained in this catalog, which adds a new item in the Me Area, which is used to adapt SAP Fiori apps at runtime.

### 7.5 Summary

In this chapter, you learned how to develop a plug-in using SAP Web IDE full-stack version and how to deploy it to both cloud and on-premise environments. When you want to call services at startup or extend SAP Fiori launchpad in your system, you can write the code you used in Chapter 4 and Chapter 5 in the plug-ins.

After reading all seven chapters of this book, I hope you’ve found that SAP Fiori launchpad is a good friend, there to give you a lot of help when you develop your SAPUI5 apps and ready for you to add your own customizations and extensions.
Contents

Preface ........................................................................................................................ 13

1 Overview of SAP Fiori Launchpad ............................................................................. 17

1.1 Introduction to SAP Fiori Launchpad ................................................................. 17
  1.1.1 End User’s Perspective .................................................................................. 18
  1.1.2 Administrator’s Perspective ......................................................................... 18
  1.1.3 Developer’s Perspective .............................................................................. 19

1.2 Versions of SAP Fiori Launchpad .................................................................... 19
  1.2.1 SAP Fiori Launchpad for SAP NetWeaver with SAP_UI Component .... 19
  1.2.2 SAP Fiori Launchpad in UI Add-On for SAP NetWeaver ....................... 20
  1.2.3 SAP Fiori Launchpad for SAP Cloud Platform ........................................ 20
  1.2.4 SAP Fiori Launchpad for SAP S/4HANA Cloud ...................................... 21
  1.2.5 SAP Fiori Launchpad for SAP HANA XS Advanced ............................. 21
  1.2.6 SAP Fiori Launchpad for SAP Enterprise Portal .................................... 21

1.3 Development Capabilities of SAP Fiori Launchpad ........................................ 21
  1.3.1 Embed SAPUI5 Applications in SAP Fiori Launchpad ............................ 22
  1.3.2 Client-Side Services ..................................................................................... 22
  1.3.3 Extend SAP Fiori Launchpad .................................................................... 22
  1.3.4 Custom Tile Types for SAP Fiori Launchpad ......................................... 23
  1.3.5 Plug-ins for SAP Fiori Launchpad ............................................................... 23

1.4 Summary ............................................................................................................. 23

2 Development Environment Setup ............................................................................. 25

2.1 Cloud-Based Development Environment ......................................................... 25
  2.1.1 Register SAP Cloud Platform Trial Account ............................................. 26
  2.1.2 Open SAP Web IDE Full-Stack Version .................................................... 29
  2.1.3 Activate SAP Cloud Platform Portal Service ........................................... 31
Index

Destination ................................................................. 93
configuration .......................................................... 94
Developer license ...................................................... 69
Developers ................................................................. 19
Development environment ........................................... 25
cloud-based ......................................................... 25
connect cloud to on-premise ....................................... 80
hardware requirements ............................................. 40
on-premise ............................................................... 38
on-premise architecture .............................................. 39
setup ........................................................................... 25
Development overview ............................................... 21
DNS record ............................................................... 64
Dynamic tiles ............................................................ 179

E
End users ..................................................................... 18
Error message ............................................................ 118
Event handler ............................................................ 133, 160, 163, 181, 196
Event listener ............................................................. 198, 212
Events ........................................................................... 133
Extensions ..................................................................... 22, 189, 274
launch page .............................................................. 211
managing element states .......................................... 210
Me Area ....................................................................... 218
options ........................................................................ 189
shell header .............................................................. 201

F
FeedContent control .................................................... 231
Fetching data .............................................................. 223
Firewall ....................................................................... 53
Footer bar ................................................................. 214
add .......................................................................... 215
Footers ....................................................................... 155, 170, 270
Frame types ............................................................... 228

G
Generic tiles ................................................................ 227
create ......................................................................... 229
GenericTile control ...................................................... 230, 247

H
Hardware key .............................................................. 73
Header end item .......................................................... 190
add ........................................................................... 205–206
pop-up ....................................................................... 207
Header item ............................................................... 190, 204
add ........................................................................... 206, 208
pop-up ....................................................................... 209
show .......................................................................... 205
with link ..................................................................... 208
Headers ........................................................................ 274
Hierarchy navigation .................................................. 199
Hook method .............................................................. 246
Hosts file ................................................................. 68
HTML5 app name ...................................................... 238
HTML5 application repository .................................... 25
HTTP Service Management app ................................ 78
HTTPS certificate ...................................................... 75
HTTPS protocol ........................................................ 87

I
Icon control ............................................................... 169
Icons .......................................................................... 286
ImageContent control .................................................. 232
Inbound navigation ..................................................... 109
index.html ................................................................. 132, 152, 169, 181
Intent .......................................................................... 19, 101, 114, 118, 121
intent ........................................................................... 97
input control ............................................................. 132, 152, 169, 181
Intent .......................................................................... 101
access ......................................................................... 101
app descriptor ........................................................... 123
availability ............................................................... 117
configuration ........................................................... 111
navigation ............................................................... 117
parameters .............................................................. 108
propose ..................................................................... 107
URL ............................................................................ 112

M
Me Area ................................................................. 18, 98, 191, 204, 218, 221, 270
add button ............................................................... 220
add settings ............................................................. 221, 223
event handlers .......................................................... 219
settings ................................................................. 220
Memory size ............................................................. 46
Microchart controls .................................................... 233
Microservices .......................................................... 21
Multiple code executions ........................................... 282, 284

N
Namespace ............................................................... 252
nano ................................................................. 58, 63
install ....................................................................... 58
Navigation ............................................................... 112, 118, 247
configuring targets .................................................... 121
flow ........................................................................... 120
services ..................................................................... 115
target ................................................................. 113, 125
testing .................................................................... 117
to previous app ....................................................... 119

J
Java Platform, Standard Edition Development .................................................. 80, 82
JavaScript APIs .......................................................... 99, 127
JavaScript promise objects ........................................ 145
jQuery object ............................................................ 222
JSON model ............................................................. 150, 157, 168, 171, 180,
245–246, 265
JSON object ............................................................. 115, 204, 211, 221, 250

O
OData service ........................................................... 127
OData service call ..................................................... 248
oninit method ........................................................ 250
onPress method ....................................................... 112
onTilePress method .................................................. 247
openSUSE Linux ....................................................... 39
download ............................................................... 41
install ................................................................. 49
Operating system ..................................................... 49
change hosts file ....................................................... 68
network settings ...................................................... 62
prepare for ABAP ..................................................... 55
Outbound navigation target ....................................... 122
Outline structure ...................................................... 153

P
Package selection ..................................................... 254
Page builder ............................................................ 251
Page control ........................................................... 98
Partition settings ..................................................... 51
Password ............................................................... 85
Personalization service .......................................... 22, 179
complex data .......................................................... 185
fetch data .............................................................. 183
initialization with container mode ......................... 186
initialize ............................................................... 182
restrict usage ......................................................... 187
save data ............................................................... 183
Plug-ins ................................................................. 23, 269
add button ............................................................. 270
add header ............................................................ 274
basic information ..................................................... 270
characteristics ......................................................... 269
classification .......................................................... 289
development .......................................................... 270
generated component .......................................... 269
parameters ............................................................ 285
predefined ............................................................. 295
SAP Cloud Platform deployment ......................... 277
SAP NetWeaver AS ABAP deployment .................. 287
<table>
<thead>
<tr>
<th>uuidd</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>check commands</td>
<td>62</td>
</tr>
<tr>
<td>check service</td>
<td>61</td>
</tr>
<tr>
<td>install</td>
<td>56</td>
</tr>
<tr>
<td>Virtual port</td>
<td>89</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>39, 59</td>
</tr>
<tr>
<td>download</td>
<td>40</td>
</tr>
<tr>
<td>install</td>
<td>45</td>
</tr>
<tr>
<td>Virtual machine set up</td>
<td>46</td>
</tr>
<tr>
<td>Web Dynpro</td>
<td>17, 97, 256</td>
</tr>
<tr>
<td>Virtual host</td>
<td>88</td>
</tr>
<tr>
<td>Virtual machine</td>
<td>45</td>
</tr>
<tr>
<td>network settings</td>
<td>67</td>
</tr>
<tr>
<td>reboot</td>
<td>61</td>
</tr>
<tr>
<td>start</td>
<td>49</td>
</tr>
<tr>
<td>Yast UI</td>
<td>56</td>
</tr>
</tbody>
</table>

VBox ....................................................... 202, 212, 219
control ............................................. 194
Versions .................................................. 19
Views ..................................................... 131
create new ............................................. 259
information .......................................... 260
Virtual hard disk ......................... 47
size ...................................................... 48
Virtual host ...................... 88
Virtual machine ..................... 45
network settings .................. 67
reboot ................................................. 61
start ................................................. 49

Web Dynpro .............................................. 17, 97, 256

XML view ................................................. 177
XTerm ..................................................... 62

Yast UI .................................................. 56
Steve Guo is an SAP mentor, developer, and trainer who has been working with SAP for more than 8 years. Since 2014 he has been working with SAP’s new technologies, including SAP HANA and SAP Fiori, and has helped numerous developers and SAP customers adopt and use SAP software. Steve is the author of three SAP training courses for SAP Fiori and other SAP user interface technologies. He has been recognized as a Gold trainer for his training work by SAP Greater China.