Getting Started with SAPUI5
# Table of Contents

1. **Introduction.** ................................................................. 3

2. **Getting Started with SAPUI5.** ........................................... 4

3. **Choose Your Browser.** ................................................... 6
   3.1 Browser Support: SAPUI5 for Desktop. ............................... 6
       3.1.1 Degradations by Feature. ........................................... 7
       3.1.2 Degradations by Control. ........................................... 9
   3.2 Browser Support: SAPUI5 for Mobile. ............................... 12

4. **Create your First Mobile App using SAPUI5.** ......................... 15
   4.1 Create a HTML Page for Your Mobile App. ........................... 15
   4.2 Initialize the Mobile App. ................................................ 16
   4.3 Add Content Pages. ....................................................... 17
   4.4 Run Your First Mobile App. ............................................. 18

5. **Create Your First SAPUI5 Application.** ................................ 21

6. **Develop Your First Application using SAPUI5 Tools.** ................. 25
   6.1 Create an SAPUI5 Application Project. ................................ 26
   6.2 Add a Control to Your View. ............................................ 29
   6.3 Implement a Method in the Controller. ................................ 30
   6.4 Create an Additional View. .............................................. 30
   6.5 Integrate a New View. .................................................. 33
   6.6 JavaScript Code Completion. ............................................ 33
   6.7 Linking your Eclipse Editor to the Demo Kit. .......................... 35
   6.8 Use JavaScript Templates. ............................................... 36
   6.9 SAPUI5 Snippets. ....................................................... 37

7. **Testing SAPUI5 Applications in Eclipse.** ............................. 45
   7.1 Test in SAPUI5 Application Preview. ................................... 45
   7.2 Test Your SAPUI5 Application on a Java Web Server. ................ 46
       7.2.1 Set up Tomcat to test SAPUI5 applications. ....................... 46
       7.2.2 Use a SimpleProxyServlet for Testing to Avoid Cross-domain Requests. .................... 47
1 Introduction

The SAPUI5 runtime is a client-side HTML5 rendering library with a rich set of standard and extension controls. It provides a lightweight programming model for desktop as well as mobile applications. Based on JavaScript, it supports RIA like client-side features. SAPUI5 complies with OpenAjax and can be used together with standard JavaScript libraries.

SAPUI5 SDK

The SDK comes together with a Demo Kit, which contains:

- A Developer Guide with a summary of valuable information around the used programming languages, open source technologies, development tools and APIs
- A Controls section containing running demo examples with descriptions and source codes
- API Reference with JavaScript documentation of Framework and Control API
- Test Suite which shows all controls running with different property settings where you can interactively adapt the controls you use for your test purpose.

Note

You can access the Demo Kit online: SAPUI5 Demo Kit on SAP HANA Cloud

This might not be the version of your local installation.

Platform Specific Help for SAPUI5

SAPUI5 is delivered to different platforms. This SAPUI5 Developer Guide focusses on platform independent topics. To find information on SAPUI5 functions specific to a platform, please refer to the respective guide:

- SAP HANA Cloud Platform
  This guide provides additional help on how to run your SAPUI5 applications on an SAP HANA Cloud platform.
- SAP HANA Platform
  This guide provides additional help on installing and testing in an SAP HANA platform environment.
- User Interface Add-On for SAP NetWeaver
  This guide provides additional help on using the SAPUI5 repository and translation infrastructure on an SAP NetWeaver application server ABAP.
2  Getting Started with SAPUI5

Here you find information on how to adjust your environment and to get started with some easy examples including an introduction to the SAPUI5 tools.

Installation Prerequisites

To find the appropriate installation option, refer to SAP note 1747308 on installation of SAPUI5.

When having installed SAP HANA Cloud Tools, SAPUI5 runtime and tools are included and ready to run.

Choose your browser

Before you really get started, check the list of supported browsers to choose one that fits your needs.

Create Your First SAPUI5 Applications

To get started with SAPUI5, you now have two different options: Either you just create an HTML page (even with notepad, if you want to) or you start directly using the SAPUI5 tools in Eclipse.

Creating Simple SAPUI5 Applications

The easiest way to work with SAPUI5 is to include a set of JavaScript libraries into your HTML page. After that you can use all controls provided by these libraries to construct one or more control trees and include them into your HTML page. The framework also supports the JavaScript Object Notation (JSON) to initialize controls with a reduced typing effort.

Creating Mobile Apps with SAPUI5

SAPUI5 provides an additional control library called sap.m, which is optimized for mobile devices.

Create Applications using SAPUI5 tools

Using the SAPUI5 application tools in Eclipse allows you to create sophisticated SAPUI5 application projects based on the Model View Controller concept. The tools provide additional features such as SAPUI5 Snippets or JavaScript Templates. They provide wizards to create SAPUI5 applications for desktop as well as for mobile devices.

Testing your SAPUI5 applications

Depending on the environment your application is running on, you can use different options to deploy and run your SAPUI5 applications. But you can always test locally in your Eclipse.

Related Information
Choose Your Browser [page 6]
You need to check which browser suits best to your use case.

Create Your First SAPUI5 Application [page 21]
You can include a bootstrap to the SAPUI5 JavaScript libraries to use SAPUI5 in an HTML page without having set up the SAPUI5 application development tools in Eclipse.

Create your First Mobile App using SAPUI5 [page 15]
Whether you develop a SAPUI5 application for desktop or for mobile, the concepts are very similar. SAPUI5 provides an additional control library called sap.m, which is optimized for mobile devices.

Develop Your First Application using SAPUI5 Tools [page 25]
The SAPUI5 application development tools in Eclipse support you in developing web applications according to the Model View Controller concept (MVC).

Testing SAPUI5 Applications in Eclipse [page 45]
You have different options to test your applications locally in Eclipse.
3 Choose Your Browser

You need to check which browser suits best to your use case.

Supported Browsers

For displaying Mobile apps for testing purposes locally on your Desktop computer, you need a WebKit based browser such as Google Chrome.

Related Information

Browser Support: SAPUI5 for Desktop [page 6]
The UI development toolkit for HTML5 (SAPUI5) is a control library based on CSS3, HTML5 and the new JavaScript API. That’s why only browsers with HTML5 capabilities are supported.

Browser Support: SAPUI5 for Mobile [page 12]

3.1 Browser Support: SAPUI5 for Desktop

The UI development toolkit for HTML5 (SAPUI5) is a control library based on CSS3, HTML5 and the new JavaScript API. That’s why only browsers with HTML5 capabilities are supported.

Browser Support for Core and Standard Libraries

Note

Also refer to SAP note 1716423

Supported Browsers

The following browsers are supported for Microsoft Windows platforms only:

- Microsoft Internet Explorer 9 and upwards, so including Microsoft Internet Explorer 10
- Mozilla Firefox 17 (aka Firefox Extended Support Release - ESR) and latest version
- Google Chrome latest version

The following browser is only supported for MAC OS X:

- Safari 5.1 and upwards
Browsers with Restricted Support

Internet Explorer 8 (IE8): There are degradations in visual design and over time also restricted functionality. For more information, see

- Degradations by Feature.
- Degradations by Control [page 9]

Not supported Browsers

Internet Explorer 6 and 7 are not supported. All browsers not mentioned above are also not supported.

Browser Support for VIZ Charting

The VIZ charting library (sap.viz) relies on the open source component D3 which in turn relies on the availability of Scalable Vector Graphics (SVG). As SVG is not supported by IE8 and not fully supported by FF ESR, the VIZ charting library is also not supported on those browsers.

3.1.1 Degradations by Feature

The following sections describe the degradations for older Browser versions, such as FireFox 3.6 or Internet Explorer 8 (IE8).

Rounded Corners

IE8 and lower versions of IE do not support the CSS property border-radius. As a result, controls in themes that use this property have square corners instead of rounded corners.

In the UX theme this affects the following controls:

- Button
- ComboBox
- DatePicker and its popup
- DropdownBox
- ListBox
- MessageBar
- Panel
- ProgressIndicator
Box Shadow

IE 8 and lower versions of IE do not support the CSS property `box-shadow`. As a result, controls in themes that use this property do not have a shadow. This property is usually used to enhance a 3D effect, for example, along vertical edges or to drop a shadow for a box "floating" above other content, which is used for all kinds of popup windows.

In the UX theme, this affects the following controls:

- Button (3D)
- ComboBox (Popup)
- DatePicker (Popup)
- DropdownBox (Popup)
- Menu (Popup)
- MenuButton (Popup)
- MessageBar (+Popup)
- ProgressIndicator (3D)
- RichTooltip
- TextArea (3D)
- TextField (3D)
- Toolbar (OverflowMenu)

Background Size

IE 8 and lower versions of IE do not support the property `background-size` which allows to stretch a background image or gradient.

This affects the Button control.
Gradient Backgrounds

IE8 only supports gradients in a limited way, so some gradients may be missing. This affects tables. In the example, compare the table header and row selectors.

Native Scrollbars

For native scrollbars, see the ComboBox example in the Box Shadow section.

3.1.2 Degradations by Control

The following table gives an overview of degradations.

<table>
<thead>
<tr>
<th>Control</th>
<th>Corners</th>
<th>Shadow</th>
<th>BG Size</th>
<th>Gradients</th>
<th>Scrollbars</th>
<th>Degradation in IE8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accordion</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No rounded corners, vertical edges without 3D effect and background has no scale when button size differs from default height</td>
</tr>
<tr>
<td>Button</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>Corners</td>
<td>Shadow</td>
<td>BG Size</td>
<td>Gradients</td>
<td>Scrollbars</td>
<td>Degradation in IE8</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CheckBox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ComboBox</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>No rounded corners, no 3D effect for vertical edges and popup has no shadow</td>
</tr>
<tr>
<td>DatePicker</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>No rounded corners, no 3D effect for vertical edges and popup has no shadow</td>
</tr>
<tr>
<td>Dialog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>Native scrollbars</td>
</tr>
<tr>
<td>DropdownBox</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>No rounded corners, no 3D effect for vertical edges and popup has no shadow</td>
</tr>
<tr>
<td>FileUploader</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Inherited from the components used (InputField and Button)</td>
</tr>
<tr>
<td>HorizontalDivider</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ListBox</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>No rounded corners, no 3D effect for vertical edges</td>
</tr>
<tr>
<td>Menu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>Popup without shadow</td>
</tr>
<tr>
<td>MenuButton</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>No rounded corners, vertical edges without 3D effect, background does not scale when button size differs from</td>
</tr>
<tr>
<td>Control</td>
<td>Corners</td>
<td>Shadow</td>
<td>BG Size</td>
<td>Gradients</td>
<td>Scrollbars</td>
<td>Degradation in IE8</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>default height, popup without shadow</td>
</tr>
<tr>
<td>MenuBar</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Menu items without rounded corners</td>
</tr>
<tr>
<td>MessageBar</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>No rounded corners, no shadow, message list popup without shadow</td>
</tr>
<tr>
<td>MessageBox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panel</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Native scrollbars</td>
</tr>
<tr>
<td>ProgressIndicator</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>No rounded corners, vertical edges without 3D effect</td>
</tr>
<tr>
<td>RadioButton</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No shadow; if it had a shadow, the shadow would not have rounded corners</td>
</tr>
<tr>
<td>RichTooltip</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slider</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Splitter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>Native scrollbars, headers/line markers only show 2-colors instead of gradient</td>
</tr>
<tr>
<td>TextArea</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>No rounded corners, vertical edges without 3D effect</td>
</tr>
<tr>
<td>TextField</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>No rounded corners, vertical edges without 3D effect</td>
</tr>
<tr>
<td>TextView</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following tables give an overview of the platforms supported by the sap.m library of SAPUI5 for Mobile. Depending on the theme you use - SAP Mobile Visual Identity or SAP Blue Crystal - different platforms and browsers are supported.

### iOS

iOS is supported as of platform version 5.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Safari</th>
<th>Web View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported with SAP Blue Crystal</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supported with SAP Mobile Visual Identity</td>
<td>Yes (iOS5 and iOS6)</td>
<td>Yes (iOS5 and iOS6)</td>
</tr>
</tbody>
</table>

Chrome and Opera are not supported. When using SAP Mobile Visual Identity, iOS7 is not supported.

### Android

Android is supported as of platform version 2.3

<table>
<thead>
<tr>
<th>Browser</th>
<th>Android Browser</th>
<th>Chrome</th>
<th>Web View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported with theme:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Blue Crystal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SAP Mobile Visual Identity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Opera, Opera mini and Firefox are not supported on Android devices.

### BlackBerry

Blackberry is supported as of platform version 10.
The `sap.makit` library supports the same platforms as *Mobile Visual Identity* theme excluding BlackBerry 10.

<table>
<thead>
<tr>
<th>Browser</th>
<th>BlackBerry Browser</th>
<th>Web View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported for both themes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Theme style when using <code>SAP Mobile visual Identity</code></td>
<td>Android</td>
<td>Android</td>
</tr>
</tbody>
</table>

Opera Mini is not supported on BlackBerry devices.

**Windows Mobile**

Windows Mobile is not supported.

**Windows Desktop**

When using Windows running on a desktop device, there is no touch support.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Internet Explorer</th>
<th>FireFox</th>
<th>Chrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported versions</td>
<td>9 or higher</td>
<td>latest and Extended Support Release</td>
<td>latest version</td>
</tr>
</tbody>
</table>

Safari and Opera are not supported.

SAPUI5 for Mobile is not supported on the desktop for *SAP Mobile Visual Identity*. Only for development scenarios you can use it with Safari and Google Chrome, with either Android or iOS style, depending on your configuration.

**Mac OS**

On Mac OS, Safari browser is supported in version 5.1 or higher when using *SAP Blue Crystal* theme.

*SAP Mobile Visual Identity* is not supported, you can use it only for development scenarios, also with Safari browser.

**Exceptions**

- *SAP Mobile Visual Identity* is *only supported* for the following controls: ActionSheet, ActionListItem, App, BusyDialog, BusyIndication, Button, CheckBox, Carousel, CustomListItem, DateTimeInput, Dialog.
● The control FlexBox is not supported in Internet Explorer 9.
● The control Carousel is not supported in Internet Explorer.
4 Create your First Mobile App using SAPUI5

Whether you develop a SAPUI5 application for desktop or for mobile, the concepts are very similar. SAPUI5 provides an additional control library called sap.m, which is optimized for mobile devices.

The sap.m library provides a mobile-style theme, which makes users feel comfortable on both, Android and Apple devices. It has a focus on touch interactions.

Note

The sap.m mobile library is optimized for mobile browsers based on WebKit. Currently, the library does not run properly on other browsers, such as Microsoft Internet Explorer and Mozilla Firefox. Therefore we recommend to use Google Chrome or Apple Safari for tests on desktop PCs.

Also, the mobile library only uses touch events and deduces its appearance from the platform it is running on. For tests with Chrome or Safari on desktop PCs, you therefore need the to set one of the following URL parameters:

- sap-ui-xx-fakeOS=android
- sap-ui-xx-fakeOS=ios
- sap-ui-xx-fakeOS=blackberry
- sap-ui-xx-fakeOS=winphone

The URL parameter suggests to emulate touch events from mouse events and to apply the styles for either Android, iOS, BlackBerry or Windows Phone. You can also add this parameter as data-sap-ui-xx-fakeOS attribute to the bootstrap script tag.

Related Information

Create a HTML Page for Your Mobile App [page 15]

You first create a HTML page and define the meta tags, a script tag to load the SAPUI5 libraries and a placeholder for your mobile app.

Initialize the Mobile App [page 16]

The sap.m library provides a control called App which is meant to be the root control of a mobile application. It provides the initialization of the HTML page, sets some meta tags to ensure an as-native-as-possible look&feel, and can manage different pages and the animations between them.

Add Content Pages [page 17]

Typical mobile applications are often composed of a number of pages/views/screens between which the user can navigate. You now add two of them to your app.

Run Your First Mobile App [page 18]

To test a mobile app on your desktop, you have to ensure some prerequisites.

4.1 Create a HTML Page for Your Mobile App

You first create a HTML page and define the meta tags, a script tag to load the SAPUI5 libraries and a placeholder for your mobile app.
1. Create a HTML page called `mobile.html`

2. Add the HTML5 doctype definition: ```<!DOCTYPE html>``` in the first line and the Internet Explorer-specific meta tag: ```<meta http-equiv="X-UA-Compatible" content="IE=edge" />``` are the beginning of the `<head>` element.
   This ensures that all browsers use the latest version of their rendering engine. Although Microsoft Internet Explorer is not really used widely on mobile devices and not yet supported by the SAPUI5 mobile library, this meta tag makes the page more future-proof.

3. Add a second meta tag: ```<meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>```.
   This lets all browsers treat the file as UTF-8 encoded (assuming that you use this encoding when editing/saving the file)

4. Add a `<div>` element to `<body>`.

5. The "sapUiBody" class should always be added to the `<body>` tag to initialize font and colors for the whole page:
   ```
   <body class="sapUiBody">
     <!-- This is where the App will live: -->
     <div id="content"></div>
   </body>
   ```

6. To load the SAPUI5 JavaScript file, that contains the library, add the following script tag in the `<head>`:
   ```
       id= "sap-ui-bootstrap"
       data-sap-ui-libs= "sap.m"
       data-sap-ui-theme= "sap_mvi">
   </script>
   ```
   Note that you are only loading the "sap.m" control library and the "sap_mvi" theme. mvi stands for Mobile Visual Identity and is the name of the SAP Mobile design.

7. Replace `<server>` and `<port>` with your local SAPUI5 installation or point to the SAPUI5 libraries on SAP HANA Cloud: `https://sapui5.hana.ondemand.com/resources/sap-ui-core.js`
   At this point, SAPUI5 including the mobile controls is loaded and ready to use.

   Initialize the Mobile App [page 16]

### 4.2 Initialize the Mobile App

The sap.m library provides a control called `App` which is meant to be the root control of a mobile application. It provides the initialization of the HTML page, sets some meta tags to ensure an as-native-as-possible look&feel, and can manage different pages and the animations between them.

Create the control and define the page that you want to display first:

```javascript
// create a mobile App
// it initializes the HTML page for mobile use and provides animated page handling
var app = new sap.m.App("myApp", {initialPage:"page1"});
// page1 should be displayed first
```
Instead of using the App control, you can also use jQuery.sap.initMobile() to set up the HTML and use other full screen controls, such as sap.m.Page or sap.m.Carousel as root element of your app.

4.3 Add Content Pages

Typical mobile applications are often composed of a number of pages/views/screens between which the user can navigate. You now add two of them to your app.

1. One sap.m.Page control is created, its title is set and the content is just one button:

   ```javascript
   // create the first page of your application
   var page1 = new sap.m.Page("page1", {
       title: "Initial Page",
       content : new sap.m.Button({
           // content is just one Button
           text : "Go to Page 2",
           tap : function() {
               app.to("page2");
               // when tapped, it triggers drilldown to page 2
           }
       })
   });
   ```

When the Button is pressed, it triggers a drilldown navigation by calling app.to("page2"). where page2 is the ID of the second page. You could also give the animation type. The default is a slide animation from right to left.

sap.m.Page controls can be used as pages, and the aggregation is called pages, but other controls could be used as well.

2. Add the following to the <script> section of the HTML page below the part, where you’ve initialized the app:

   ```javascript
   // create the second page of your application
   var page2 = new sap.m.Page("page2", {
       title: "Page 2",
       showNavButton: true,
       // page 2 should display a back button
       navButtonTap: function() {
           app.back();
           // when tapped, the back button should navigate back up to page 1
           content : new sap.m.Text({text:"Hello Mobile World!"})
       }
   });
   ```

showNavButton is set to true to get a Back button displayed. When this button is triggered, the handler calls app.back(). This causes an inverse animation, which leads back to the main page.

A header icon, which is only visible on Android, and "Hello Mobile World" content is also given.

3. Finally, add the two pages to the App:

   ```javascript
   // add both pages to the App
   app.addPage(page1).addPage(page2);
   ```

The app is placed into the HTML like a SAPUI5 desktop control. The App takes care to cover the whole screen.

*Run Your First Mobile App* [page 18]
4.4 Run Your First Mobile App

To test a mobile app on your desktop, you have to ensure some prerequisites.

Your operating system is Microsoft Windows and you use a current WebKit-based browser (Google Chrome or Apple Safari). Other operating systems work as well, but the procedure may differ. Also, the mobile library is currently not optimized for Mozilla Firefox and Microsoft Internet Explorer.

Note

The URL in the script tag is pre-filled as `https://sapui5.hana.ondemand.com/resources/sap-ui-core.js`. This is the URL where the resources are located in the SAP HANA Cloud delivery. Test this URL first and if it does not work, replace this URL with the location of SAPUI5 on your local server.

Also note that the version of SAPUI5 deployed on `https://sapui5.hana.ondemand.com/` may be updated with a delay of some days or weeks after a new release of SAPUI5, even though we try to keep them in sync. This example will work nevertheless.

2. Enter a name for the new file, for example "mobile.html" and confirm the extension change warning.
3. Right-click on the new file and choose Edit. Make sure it opens in Notepad and not in MS Word.
4. Copy and paste the HTML code below and save the file. Keep in mind that the SAPUI5 URL may need to be adapted.
5. Drag and drop the file into the browser window.
6. To load the example on a mobile device, you put the file on a server.
7. To play around with the app in your desktop browser, add the following URL parameter to the file URL: `sap-ui-xx-fakeOS=ios`, so that the URL reads: "mobile.html?sap-ui-xx-fakeOS=ios". This enables the simulation of touch events on desktop PCs. This also enables the iPhone/iPad styling; if you want to see the Android styling, use `sap-ui-xx-fakeOS=android` instead.

```html
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
<title>Mobile App in 23 Seconds Example</title>
<script src="https://sapui5.hana.ondemand.com/resources/sap-ui-core.js"
    id="sap-ui-bootstrap"
    data-sap-ui-libs="sap.m"
    data-sap-ui-theme="sap_mvi">
</script>

// only load the mobile lib "sap.m" and the "sap_mvi" theme -->

// create a mobile App
// it initializes the HTML page for mobile use and provides animated page handling
var app = new sap.m.App("myApp", {initialPage:"pagel"}); // pagel should be displayed first

// create the first page of your application
var pagel = new sap.m.Page("pagel", { title: "Initial Page",
    content : new sap.m.Button({ // content is just one Button
text : "Go to Page 2",
tap : function() {
            app.to("page2"); // when tapped, it triggers drilldown to page
```
// create the second page of your application
var page2 = new sap.m.Page("page2", {
    title: "Page 2", showNavButton: true, // page 2 should display a back button
    navButtonTap: function(){
        // when tapped, the back button should navigate back up to page 1
        app.back();
    },
    content : new sap.m.Text({text:"Hello Mobile World!"})
});
app.addPage(page1).addPage(page2); // add both pages to the App
// place the App into the HTML document
app.placeAt("content");
If you have set the sap-ui-xx-fakeOS URL parameter, you can navigate to the second page by clicking the button.

To open the application on a real mobile device, you can also put the HTML document on a Web server and load the resulting URL in your mobile browser.
5 Create Your First SAPUI5 Application

You can include a bootstrap to the SAPUI5 JavaScript libraries to use SAPUI5 in an HTML page without having set up the SAPUI5 application development tools in Eclipse.

This page explains how to create and run a simple SAPUI5 application from scratch within twenty seconds (with some practice... the current record is 16 seconds).

If you are interested in exactly doing this without reading too much, you can skip the background information and read the And how to do it in 20 Seconds section below.

Background Information

As SAPUI5 is a client-side web UI library meaning that it runs in a browser, a SAPUI5 application typically is composed of an HTML page and, if required, many more files.

SAPUI5 is implemented in JavaScript. For loading SAPUI5, its bootstrap needs to be included with a <script> tag. The last two attributes select the visual design to apply initially (other choices would be sap_hcb or sap_platinum) and the SAPUI5 control library/libraries to use (sap.ui.dev would be another one). In your scenario you need to make sure the URL points to a SAPUI5 installation.

```html
<script id="sap-ui-bootstrap"
    src="resources/sap-ui-core.js"
    data-sap-ui-theme="sap_goldreflection"
    data-sap-ui-libs="sap.ui.commons"></script>
```

SAPUI5 UI elements are created and modified programmatically:

```javascript
// create the button instance
var myButton = new sap.ui.commons.Button("btn");

// set properties, e.g. the text (there is also a shorter way of setting several properties)
myButton.setText("Hello World!");

// attach an action to the button's "press" event (use jQuery to fade out the button)
myButton.attachPress(function(){$('#btn').fadeOut()});
```

There is also a shorthand notation based on JSON for setting multiple properties; you could also write:

```javascript
var myButton = new sap.ui.commons.Button({text:"Hello World!",tooltip:"Hello Tooltip!"});
```

Finally you need to tell SAPUI5 where the UI control should be placed. You can just give the ID of an element in the page to do so:

```javascript
// place the button into the HTML element defined below
myButton.placeAt("uiArea");
```

This element must exist somewhere in the HTML page, so you need to put the following code to the desired place within the <body>:

```html
<div id="uiArea"></div>
```
Currently, you can only put one SAPUI5 control into a parent; for adding more SAPUI5 controls you need to either define more parents, or use a SAPUI5 layout control which can arrange many children.

An alternative way to create and initialize the control in a more jQuery-style manner is also available:

```javascript
$(function(){
    $('uiArea').sapui('Button', 'btn', {
        text:'Hello World!'
        press:function(){$('btn').fadeOut();}
    });
});
```

As a minor detail, the `<body>` should have a certain CSS class, so the page background and some other styles are properly set:

```html
<body class="sapUiBody">
```

There are two meta tags at the beginning of the `<head>`: The first meta tag is used to ensure that Internet Explorer 8+ uses its most standard-compliant rendering mode. The second meta tag is used to let all browsers treat the file as UTF-8 encoded (assuming that you use this encoding when editing/saving the file):

```html
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```

### And How to do it in 20 Seconds

Assumption for these instructions to work exactly as described: You have a Windows Computer (other OS will work similarly), Internet Explorer 9+ with security option set to Access data across domains for the respective zone, FireFox 13+, Safari 5+ or Chrome 20+, and you know where you can refer to SAPUI5 on some server.

<i>Note</i>

The URL in the script tag is pre-filled as `https://sapui5.hana.ondemand.com/resources/sap-ui-core.js`. This is the URL where the resources are located in the SAP HANA Cloud delivery. Test this URL first and if it does not work, replace this URL with the location of SAPUI5 on your local server.

Also note that the version of SAPUI5 deployed on `https://sapui5.hana.ondemand.com/` may be updated with a delay of some days or weeks after a new release of SAPUI5. This example will work nevertheless.

Proceed as follows:

1. Right click your desktop and select <i>New</i> <i>Text Document</i>.
2. Name the new file, for example "ui5.html", and accept the extension change warning.
3. Right click the new file and select <i>Edit</i>. Make sure that the file opens in Notepad and not in MS Word.
4. Copy and paste the below HTML code. Keep in mind to adapt the SAPUI5 URL
5. Drag the file to the browser window.
6. That was it.

```html
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
</head>
```
Result

If you followed the steps above you should now see a button like this which fades out when clicked:
Next Steps

You can now do the following:

- Add more buttons.
- Let them do trickier things.
- Use a different visual theme as mentioned above, for example sap_ux
- Find out about further properties and events of button controls and use those.
- Find out about further controls and add those. For more information, see the Controls Gallery in the Demo Kit.
6  Develop Your First Application using SAPUI5 Tools

The SAPUI5 application development tools in Eclipse support you in developing web applications according to the Model View Controller concept (MVC).

The SAPUI5 application development tools for Eclipse provide wizards to support you in creating SAPUI5 applications in an easy way. With the SAPUI5 application project wizard, the necessary application skeleton containing view(s) and controller will automatically be created.

Creating SAPUI5 applications

The SAPUI5 tools support you in creating applications according to MVC. In this section we guide you through a simple example. You will create a SAPUI5 application project, which includes a control, a method in the controller and an additional view.

More Information

Create an SAPUI5 Application Project [page 26]
Add a Control to Your View [page 29]
Implement a Method in the Controller [page 30]

Utilities

With Eclipse you can make use of utilities for JavaScript development. Additionally SAPUI5 provides templates and snippets.

- JavaScript Code Completion
  The Eclipse JavaScript Development Tools (JSDT) provide an editor which parses scripts and offers code completion functionality. The core libraries for the code completion are made available automatically.

  Note
  If your Eclipse installation contains the org.eclipse.wst.jsdt.feature feature in Version 1.3.1, we recommend to update it. Invoking the JavaScript code completion in version 1.3.1 may cause Eclipse to crash.

More Information

JavaScript Code Completion [page 33]
6.1 Create an SAPUI5 Application Project

To create a SAPUI5 Application Project, you must have installed the SAPUI5 Application Development feature in your Eclipse installation.

1. Start the New SAPUI5 Application Project wizard in the Eclipse by choosing New Other SAPUI5 Application Development Application Project.

![Select a wizard image](image-url)
2. Enter the following project-related data:

- Project name
- Location (optional, prefilled from the current workspace)
- Target device: 'Desktop' or 'Mobile'
- Select Create an Initial View
  Views can also be added later using the SAPUI5 Application View wizard
3. Enter the following view-related data:

○ Choose the folder in which the view shall be created
○ Enter a unique name for your view
○ Choose the Development Paradigm.

After finishing the wizard, the system performs the following steps:

● A new dynamic web project is created. All relevant files are created in the WebContent folder.
● A prefilled index.html is created which contains sap.ui.commons lib and sap_goldreflection theme in the bootstrap in case of a desktop target device and the sap.m.lib and sap_mvi theme in case of mobile target device.
● In WEB-INF folder a web.xml file is created which contains settings for resource handling and the use of SimpleProxyServlet.
● The installed SAPUI5 UI lib plugins are automatically added to the Java build path and added to the deployment assembly.
● The SAPUI5 class path container (if available) is automatically added to the JavaScript include path.
● The index.html page is opened in the standard editor.
● Inside the JavaScript block of index.html, code completion is available, see JavaScript Code Completion.
● An automatic switch to the J2EE perspective is performed.
If you have selected the **Create an Initial View** option on the first page of the **SAPUI5 Application Project** wizard, a view and a view controller are created and the coding to call the view is added to the index.html file.

*Add a Control to Your View* [page 29]

*Implement a Method in the Controller* [page 30]

*Create an Additional View* [page 30]

*Integrate a New View* [page 33]

**Related Information**

*JavaScript Code Completion* [page 33]
When using the SAPUI5 tools, code completion is enabled automatically, without the tools, you need to enable it.

*Use JavaScript Templates* [page 36]
You can add SAPUI5 control-specific templates for JavaScript code. Such templates are available, for example, in JavaScript views of SAPUI5 application tools development.

*SAPUI5 Snippets* [page 37]

*Linking your Eclipse Editor to the Demo Kit* [page 35]
You can use Quick Fixes to display the API documentation of a SAPUI5 control in the Demo Kit.

### 6.2 Add a Control to Your View

In your SAPUI5 application project, the first step to build your application is to add a control to your view and implement a method to react on user interaction. In this case you create a button and implement a function to react when the user presses it.

To add a control to your view, add the following coding depending on the type of your view:

- **In a JS view** add the following to the `createContent` function:

  ```javascript
  var aControls = [];
  var oButton = new sap.ui.commons.Button({
    id : this.createId("MyButton"),
    text : "Hello JS View"
  });
  aControls.push(oButton.attachPress(oController.doIt));
  return aControls;
  ```

- **In an HTML view** add the following to the `template` tag:

  ```html
  <div data-sap-ui-type="sap.ui.commons.Button" id="MyButton" data-text="Hello HTML View" data-press="doIt">
  </div>
  ```

- **In an XML view** add the following coding to the `core` tag:

  ```xml
  <Button id="MyButton" text="Hello XML View" press="doIt"/>
  ```

- **In a JSON view** add the following to the `content` function:

  ```json
  "Type":"sap.ui.commons.Button",
  "id":"MyButton",
  ```
A button is added to your view with an event that is triggered when the user presses it.
The `doIt` method, which is called in each of these view types, is implemented in the controller:

Implement a Method in the Controller [page 30]

### 6.3 Implement a Method in the Controller

All functions that are not directly related to the user interface should be handled in the controller to ensure clear separation between UI and data. In this case you add a method to handle the event, which is attached to a button.

You've created a button as described in: Add a Control to Your View [page 29]

To handle this event, add the following function to the controller:

```javascript
doIt : function(oEvent) { alert(oEvent.getSource().getId() + " does it!"); }
```

### 6.4 Create an Additional View

- A SAPUI5 application view can only be created for a SAPUI5 application project that has been created with the SAPUI5 Application Wizard and **not** for other kinds of projects.
- A SAPUI5 application view name needs to be unique inside the project folder.
- The specified folder for a SAPUI5 application view needs to be `WebContent/<application name>` or a sub folder.
1. Choose > New > Other... > SAPUI5 Application Development > View to open the New SAPUI5 Application View wizard.
2. Fill in the required data:

- Select the SAPUI5 application project, in which you want to create the view.
- Select a folder, in which you want to store the view (default is `WebContent/<application name>`).
- Enter a name for the view.
- Select the development paradigm, see **View Types**.

When you finish the wizard, the system creates the view in the specified folder. The file name suffix indicates the development paradigm:

- `<viewname>.view.js` for JavaScript views
- `<viewname>.view.xml` for XML views
- `<viewname>.view.json` for JSON views
- `<viewname>.view.html` for HTML views

If the corresponding index.html file contains `sap.m.lib` in the bootstrap, that is, if the SAPUI5 application project has been created for a mobile target device, the view contains coding for instantiating a mobile page control `sap.m.Page`.

The system also creates a controller file `<viewname>.controller.js` with draft coding.

For JavaScript views, code completion is available, see **JavaScript Code Completion**.
6.5 Integrate a New View

To integrate a new view, you can either add it to index.html or nest it into another view.

If you create a new view for an existing SAPUI5 application project, the view needs to be manually called.

To call a view, choose from the following options:

- Directly embed the new view in the index.html page
- All Views can be nested independent of the view type. Each view type behaves like any SAPUI5 control. The `viewName` property defines, which views are embedded. To nest a view, proceed according to the given examples:

For XML view type:

```xml
<core:View controllerName="sap.hcm.Address" xmlns="sap.ui.commons"
xmlns:core="sap.ui.core"
xmlns:html="http://www.w3.org/1999/xhtml">
  <Panel>
    <core:JSView id="myJSView" viewName="sap.hcm.Bankaccount"/>
  </Panel>
</core:View>
```

For HTML views, the nested view looks as follows:

```html
<template data-controller-name= "example.mvc.test">
  <div data-sap-ui-type= "sap.ui.core.mvc.HTMLView" id= "MyHTMLView" data-view-name= "example.mvc.test2" ></div>
  <div data-sap-ui-type= "sap.ui.core.mvc.JSView" id= "MyJSView" data-view-name= "example.mvc.test2" ></div>
  <div data-sap-ui-type= "sap.ui.core.mvc.JSONView" id= "MyJSONView" data-view-name= "example.mvc.test2" ></div>
  <div data-sap-ui-type= "sap.ui.core.mvc.XMLView" id= "MyXMLView" data-view-name= "example.mvc.test2" ></div>
</template>
```

6.6 JavaScript Code Completion

When using the SAPUI5 tools, code completion is enabled automatically. Without the tools, you need to enable it.

- **Note**
  
  If your Eclipse installation contains the `org.eclipse.wst.jsdt.feature` feature in Version 1.3.1, we recommend to update it. Invoking the JavaScript code completion in version 1.3.1 may cause Eclipse to crash.
Automatic Code Completion for SAPUI5 Application Projects

The Eclipse JavaScript Development Tools (JSDT) provide an editor which parses scripts and offers a code completion functionality.

Code Completion for SAPUI5 Views

For JavaScript views, code completion is available.
Enabling Code Completion for Other Projects

If you are not working with a SAPUI5 application project, you can perform the following preparing steps to add the required SAPUI5 core libraries to the JavaScript include path.

Ensure that the JavaScript Facet is set and proceed as follows:

1. Open Project Properties.
2. Select Project Facets.
3. If you do not see the list of all possible facets, click the link: Convert to facet form and wait a second to see all available facets.
4. Mark JavaScript Facet on the same view.
5. Leave the project properties.

Your project now has the JavaScript facet. Now you can add the SAPUI5 core libraries. Proceed as follows:

- Open Project Properties.
- Choose JavaScript Include Path.
- Select Add JavaScript Library....
- Select SAPUI5.

You should now be able to see the following JavaScript resources in your project:

- ECMAScript Built-in Library
- ECMA 3 Browser Support Library
- WebApp
- SAPUI5 Core Libraries
  - sap.ui.commons
  - sap.ui.commons.enums
  - sap.ui.commons.layout
  - sap.ui.commons.MessageBox
  - sap.ui.core
    - AccessibleRole
    - BarColor
    - BusyIndicator
    - Configuration
    - Control
    - Core
    - CSSSize
    - Design

6.7 Linking your Eclipse Editor to the Demo Kit

You can use Quick Fixes to display the API documentation of a SAPUI5 control in the Demo Kit.
1. Place the cursor on the actual SAPUI5 control name in your JavaScript code or in your XMLView. The name of the control in JavaScript code usually starts with `sap`.

2. To see all available Quick Fixes, press `CTRL+1`.

3. To open the API documentation of the control in the Demo Kit, choose `Display in Demo Kit`.

### 6.8 Use JavaScript Templates

You can add SAPUI5 control-specific templates for JavaScript code. Such templates are available, for example, in JavaScript views of SAPUI5 application tools development.

The templates are an overview over all available

- control properties
- aggregations
- associations and
- events

To use the JavaScript templates, the SAPUI5 application development tools feature has to be installed in your Eclipse.

**Note**

If your Eclipse installation contains the feature `org.eclipse.wst.jsdt.feature` in Version 1.3.1, we recommend to update it. In this version, invoking the JavaScript code completion may cause Eclipse to crash.

1. To insert a template, open the JavaScript editor.

2. Start typing the name of the respective control or the name of the alias, for example `button`.

3. Choose `CRTL+SPACE` and choose the control from the code completion list.

All properties and events are inserted.
6.9 SAPUI5 Snippets

You can add SAPUI5-specific code parts, so called SAPUI5 Snippets. Code snippets are templates and examples on how to use the SAPUI5 runtime and controls and what can be done with them. SAPUI5 snippets are available as prepared HTML pages with no separation between model, view and, controller (MVC) and they are generated during startup of the Eclipse runtime.

Before you can use the SAPUI5 snippets, the SAPUI5 application development tools must be installed in your Eclipse.

To open the Snippets view, proceed as follows:
1. Choose **Window > Show View > Other...**
2. In the Show View dialog, choose \General \Snippets and confirm your selection with OK.

3. The Snippet view opens.

To insert a snippet, proceed as follows:

1. Create a SAPUI5 application project, see Creating a SAPUI5 Application Project.
2. Open the index.html in the HTML editor.
3. Delete all content.
4. To insert the snippet code, double click on the snippet or use drag&drop.
5. Save the code and run it in the integrated browser.
6. **Note**

If you have problems with incorrect rendered pages, open the default external browser.
The page should then be displayed correctly:
### Products

<table>
<thead>
<tr>
<th>ID</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT-1000</td>
<td>50</td>
</tr>
<tr>
<td>AP-3</td>
<td>12</td>
</tr>
<tr>
<td>AB-2000</td>
<td>10</td>
</tr>
<tr>
<td>SO-3000</td>
<td>5</td>
</tr>
<tr>
<td>MO-3000</td>
<td>15</td>
</tr>
<tr>
<td>AO-30</td>
<td>25</td>
</tr>
<tr>
<td>AP-2</td>
<td>14</td>
</tr>
</tbody>
</table>

Product description:
- Apple iPad 3

Chart showing product distribution:
7 Testing SAPUI5 Applications in Eclipse

You have different options to test your applications locally in Eclipse.

You have created a SAPUI5 application using the SAPUI5 application development tools, see Developing Applications Using SAPUI5 Tools.

1. To use SAPUI5 application preview, see Test in SAPUI5 Application Preview
2. To use a Java Web Server, see Test Your SAPUI5 Application on a Java Web Server [page 46]

If you have created a SAPUI5 application for a mobile target device, the application can only run in a WebKit-based browser. You can, however, use the mentioned options for testing your application locally in Eclipse and copy and paste the URL into a WebKit-based browser.

If your application has to access resources on a back-end system (OData), you must enable the back-end access for local testing.

7.1 Test in SAPUI5 Application Preview

You access the SAPUI5 application preview using the Web App Preview, provided with the embedded Jetty server. You can quickly check on your application and open it in the default browser.

**Note**
Jetty provides an internal web preview used in the SAPUI5 application preview. As of the Juno release of Eclipse, this features needs to be installed explicitly.

1. To test the new application with the application preview in an embedded Jetty server, right-click the HTML file or the project node and choose Run As Web App Preview. Everything is configured automatically.
2. To refresh after having changed a file of your SAPUI5 application, choose Refresh on the left hand side of the preview editor to refresh the SAPUI5 preview.
3. To check the files of your SAPUI5 application project in an external browser, choose Open in external browser on the right hand side of the preview editor. This function opens the external browser which is specified in the Eclipse preferences under General Web Browser. This is usually the default browser of your PC. For other external browsers, you can also copy the URL from the text field of the editor to the external browser.

Open in External Browser
7.2 Test Your SAPUI5 Application on a Java Web Server

To test the application in a Java Web Server environment, ensure that the corresponding Server Adapter Eclipse Plugin is installed.

To test the new application, right click the new project and choose \( \textit{Run As} \to \textit{Run on Server} \). Select a server (for example, Apache Tomcat), and choose \textit{Finish}.

Related Information

- Set up Tomcat to test SAPUI5 applications [page 46]
- Use a SimpleProxyServlet for Testing to Avoid Cross-domain Requests [page 47]

7.2.1 Set up Tomcat to test SAPUI5 applications

To test on a Java web server, you can install an Apache Tomcat.

The recommended versions of Tomcat are 6.0 or higher.

1. Download Tomcat from the respective web site. You can either install the Windows Service Installer, or download and unzip the zip file (you can find the startup script in the bin folder).

2. To deploy your application, choose one of the following options:
   - Without Eclipse
     To deploy SAPUI5 on Tomcat without Eclipse, you can set up a Tomcat user with manager privileges (see the Tomcat documentation) and use the tomcat manager application for deployment. We recommend,
however, to just move your war file into the `webapps` folder inside the tomcat directory. Tomcat takes care of the deployment, but a restart may be necessary.

If you run Tomcat without Eclipse, set the environment variable `JAVA_HOME` to JDK6

- **In Eclipse**
  To run web applications in Eclipse, you need to register the Tomcat web server.

  1. Open the Java EE perspective in Eclipse.
  2. Open the `Servers` view, for example by pressing `CTRL` + `3` and then entering `Servers`.
  3. To open the `New Server` wizard, choose `New` > `Server` from the context menu.
  4. Choose `Apache Tomcat 7.0` and add a new server runtime by pointing to the folder in which you have unzipped the Tomcat installation files. Finish the wizard. You should now see Tomcat v7.0 server in the `Servers` view.
  5. Open the configuration overview by double clicking the Tomcat v7.0 server in the `Servers` view. Select the `Serve modules without publishing` server option.

### 7.2.2 Use a SimpleProxyServlet for Testing to Avoid Cross-domain Requests

If you are testing locally in your Java Eclipse environment and you want to access an OData service in the ABAP system, a proxy is needed to ensure the same origin policy. In an SAPUI5 application project you can use a SimpleProxyServlet for local testing.

**Caution**

Be aware that due to security reasons the SimpleProxyServlet is restricted to local testing purposes only. It can only be used for local host scenarios (accessing Gateway services to avoid cross-domain issues) and will not work when deployed on an application server. For productive use, refer to a mature proxy servlet.

**Note**

If you have issues with accessing HTTPS systems via the `ResourceServlet` or the `SimpleProxyServlet` it may be necessary to import the root certificate into the Java keystore.

Ideally, all OData service URLs should be in one file to make the exchange easier - either in the `index.html`, or in one separate `.js` file which needs to be included. The application is responsible for exchanging the URLs before checking in and after checking out to SAPUI5 Repository. You can also use the helper function `getServiceUrl` for which also the application is responsible. See the following example:

```java
#!java
<script>
//var serviceUrl = "/mypath/myservice";    //url when running on the ABAP system
//var serviceUrl = "proxy/mypath/myservice";  //url when running locally in Eclipse
var serviceUrl = getServiceUrl("/mypath/myservice");
function getServiceUrl(sServiceUrl) {
  //for local testing prefix with proxy
  //if you and your team use a special host name or IP like 127.0.0.1 for localhost
  please adapt the if statement below
  if (window.location.hostname == "localhost") {
    return "proxy/mypath/myservice";
  } else {
    return sServiceUrl;
  }
}
</script>
```
As parameter for the `getServiceUrl` helper method, use the URL of the OData service document without `{protocol}:://{host name}:{port number}`, for example: `/mypath/myservice`.

**Note**
Place the script tag before the script that calls the view (`sap.ui.view`).

### Intranet Servers

The `SimpleProxyServlet` allows proxy requests to an arbitrary server in the intranet.

The proxy URL that is used in the coding looks like this: `proxy/<service url>`.

Open the `web.xml` file located in the `<WebContent folder name>/WEB-INF` folder and configure the parameter `com.sap.ui5.proxy.REMOTE_LOCATION` of the `SimpleProxyServlet` where the placeholders `{protocol}`, `{host name}`, `{port number}` are to be exchanged by the real protocol, host name and port number:

```xml
<servlet>
    <servlet-name>SimpleProxyServlet</servlet-name>
    <servlet-class>com.sap.ui5.proxy.SimpleProxyServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>SimpleProxyServlet</servlet-name>
    <url-pattern>/proxy/*</url-pattern>
</servlet-mapping>

<context-param>
    <param-name>com.sap.ui5.proxy.REMOTE_LOCATION</param-name>
    <param-value>{protocol}://{host name}:{port number}</param-value>
</context-param>
```

### Internet Servers

The `SimpleProxyServlet` can be configured for proxy requests to internet servers in the same way as for intranet servers. Additional proxy settings may be necessary.

As the `SimpleProxyServlet` automatically uses the proxy settings from your Eclipse this can be configured in Eclipse under `Window > Preferences`, and select `General > Network Connections`. Here you can specify the proxy entries and the proxy bypass.

For example, set `Active Provider` to Manual, `Schema=HTTP`, `Host=proxy`, `Port=8080` under proxy entries and localhost, `*.sap.corp` as `Host` under proxy bypass.
Simple Proxy Servlet - Restriction Regarding DELETE Requests

The simple proxy servlet currently does not support the sending of HTTP DELETE requests with content. This is due to restrictions of the Java SE functionality that is used. If an HTTP DELETE request with content is sent, an HTTP 500 result status is sent with the description: "The HttpURLConnection used by the SimpleProxyServlet doesn’t allow to send content with the HTTP method DELETE. Due to spec having content for DELETE methods is possible but the default implementation of the HttpURLConnection doesn’t allow this!"

For practical purposes, this restriction should have only minor effects. This is because:

- When applying a REST-like programming style, an HTTP DELETE request would use the URL to transmit which objects should be deleted, but not the content.
- When building your own protocol that uses the content of the HTTP request, you typically use HTTP POST.