

CAPM Binding Data to a Table in a Basic SAP UI5 Application (Chapter 1 - READ)

Authored by:

Mahesh L
SAP UI5 Fiori Associate Consultant
Email: lokkidimahesh6@gmail.com

This document provides a beginner-friendly guide to binding data to a table in a basic SAP UI5 application, focusing on integration with an SAP HANA database and deployment within the Cloud Foundry runtime environment. It will walk you through setting up your development environment, creating data models, and connecting them to your UI5 application.

What You Will Learn:

- Setting up the development environment for binding data to a UI5 table.
- Connecting an SAP HANA database with your SAP UI5 application.
- Creating an XML view control for the UI5 table.
- Binding data with OData V4 to populate the table in the UI.

Chapter 1: Implementing CRUD Operations – Read (Chapter 1)

This guide currently covers the **Read** operation (data retrieval) in the SAP UI5 table. The **Create**, **Delete**, and **Update** operations will be covered in the upcoming chapters of this series.

Prerequisites:

- Basic knowledge of JavaScript and Node.js.
- Familiarity with SAP BTP.
- Installed tools: Node.js, npm, and SAP Business Application Studio or Visual Studio Code.

For detailed steps on database and instance creation in CAPM, please refer to the following document:

https://www.linkedin.com/posts/mahesh-lokkidi-8664b9135_sap-capm-steps-pdf-guide-activity-7278389861781315585-nKxN?utm_source=share&utm_medium=member_desktop

1. Open SAP HANA Cloud

If SAP HANA Cloud is not visible in your environment, please subscribe to SAP HANA Cloud first. Once subscribed, proceed with creating a Cloud Foundry instance.

For detailed instructions and reference, kindly refer to the document linked above.

The screenshot shows the SAP BTP Cockpit interface. The left sidebar contains navigation links for Overview, Services, Cloud Foundry, HTML5 Applications, Connectivity, Security, and Legal Information. The main content area displays a message about an extended trial account, the current subaccount name (trial), and the title "Subaccount: trial - Instances and Subscriptions". Below this, there are tabs for Subscriptions (2), Instances (5), and Environments (1). The Subscriptions section lists two entries: SAP HANA Cloud (Plan: tools) and SAP Business Application Studio (Plan: trial), both marked as "Subscribed". The Instances section shows five instances created in Cloud Foundry, Kyma/Kubernetes, or Other environments. The table headers for Instances include Instance, Service, Plan, Runtime Environment, Scop, Credentials, and Status.

Application	Plan	Changed On	Status
SAP HANA Cloud	tools	23 Dec 2024	Subscribed
SAP Business Application Studio	trial	23 Dec 2024	Subscribed

Instance	Service	Plan	Runtime Environment	Scop	Credentials	Status
Instance 1	Service 1	Plan 1	Runtime Env 1	Scop 1	Cred 1	Active
Instance 2	Service 2	Plan 2	Runtime Env 2	Scop 2	Cred 2	Active
Instance 3	Service 3	Plan 3	Runtime Env 3	Scop 3	Cred 3	Active
Instance 4	Service 4	Plan 4	Runtime Env 4	Scop 4	Cred 4	Active
Instance 5	Service 5	Plan 5	Runtime Env 5	Scop 5	Cred 5	Active

2. Start SAP HANA Cloud

Click on **Start** to initiate the SAP HANA Cloud service.

Note: For trial accounts, it is required to start the service daily. However, this step is not necessary for real-time (paid) accounts, as the service remains active.

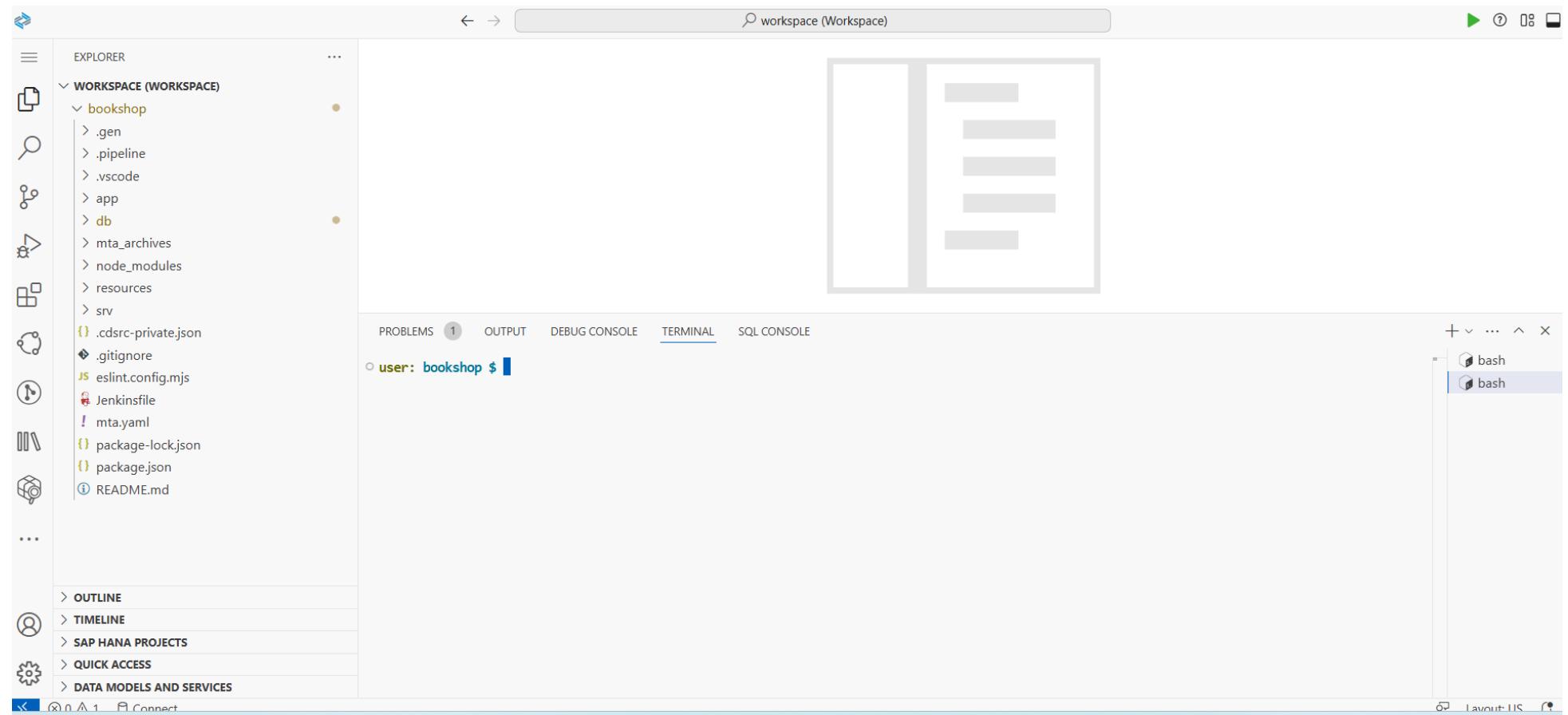
The screenshot shows the SAP HANA Cloud Central interface. On the left, there's a sidebar with icons for Home, Instances, Subaccounts, Roles, and Help. The main area is titled "Instances" and shows a table with one row for "SHDB". The columns are: State (Stopped), Name (SHDB), Type (SAP HANA Database), Notifications (1), Runtime Environment (Cloud Foundry), Memory (16 GB), Storage (80 GB), and Compute (1 vCPUs). Below the table, it says "1 node" and "0 replicas". To the right of the table, a context menu is open, listing options like "Manage Configuration", "Add Data Lake", "Copy SQL Endpoint", etc. The "Start" option is highlighted with a red box. At the bottom right of the interface, there's a "..." button.

State	Name	Type	Notifications	Runtime Environment	Memory	Storage	Compute
Stopped	SHDB	SAP HANA Database	1	Cloud Foundry	16 GB	80 GB	1 vCPUs

3. Open SAP Business Application Studio

Open the **SAP Business Application Studio**.

In this example, I have already created a **CAPM Bookshop** project. For your reference, I have provided the document link on the first page. The project I created is named **Supermarket**, which is similar to the Bookshop project.



4.Create CDS Files for Database and Service

I have created the following CDS files:

1. **schema.cds** – This file defines the database schema and tables for the application.
2. **admin-service.cds** – This service file defines the OData services for interacting with the database.

These files are essential for setting up the database structure and exposing the necessary APIs for CRUD

The screenshot shows the SAP Studio IDE interface. The left pane is the Explorer view, displaying the project structure under 'WORKSPACE (WORKSPACE)'. The 'schema.cds' file is selected and highlighted with a red border. The right pane is the Editor view, showing the content of the 'schema.cds' file. The bottom pane is the Terminal view, showing a user session for the 'bookshop' project. The code in the schema.cds file defines entities for Books and Authors.

```
bookshop > db > schema.cds > Books > genre
1 using { Currency, managed, sap } from '@sap/cds/common';
2 namespace sap.capire.bookshop;
3
4 entity Books : managed {
5   key ID : Integer;
6   title : localized String(111);
7   descr : localized String(1111);
8   author : Association to Authors;
9   genre : Association to Genres;
10  stock : Integer;
11  price : Decimal;
12  currency : Currency;
13  image : LargeBinary @Core.MediaType : 'image/png';
14 }
15
16 entity Authors : managed {
17   key ID : Integer;
18   name : String(111);
19   dateOfBirth : Date;
20   dateOfDeath : Date;
21   placeOfBirth : String;
```

TERMINAL

```
user: bookshop $
```

```

schema.cds

using { Currency, managed, sap } from '@sap/cds/common';
namespace sap.capire.bookshop;

entity Books : managed {
    key ID : Integer;
    title : localized String(111);
    descr : localized String(1111);
    author : Association to Authors;
    genre : Association to Genres;
    stock : Integer;
    price : Decimal;
    currency : Currency;
    image : LargeBinary @Core.MediaType : 'image/png';
}

entity Authors : managed {
    key ID : Integer;
    name : String(111);
    dateOfBirth : Date;
    dateOfDeath : Date;
    placeOfBirth : String;
    placeOfDeath : String;
    books : Association to many Books on books.author = $self;
}

/** Hierarchically organized Code List for Genres */
entity Genres : sap.common.CodeList {
    key ID : Integer;
    parent : Association to Genres;
    children : Composition of many Genres on children.parent = $self;
}

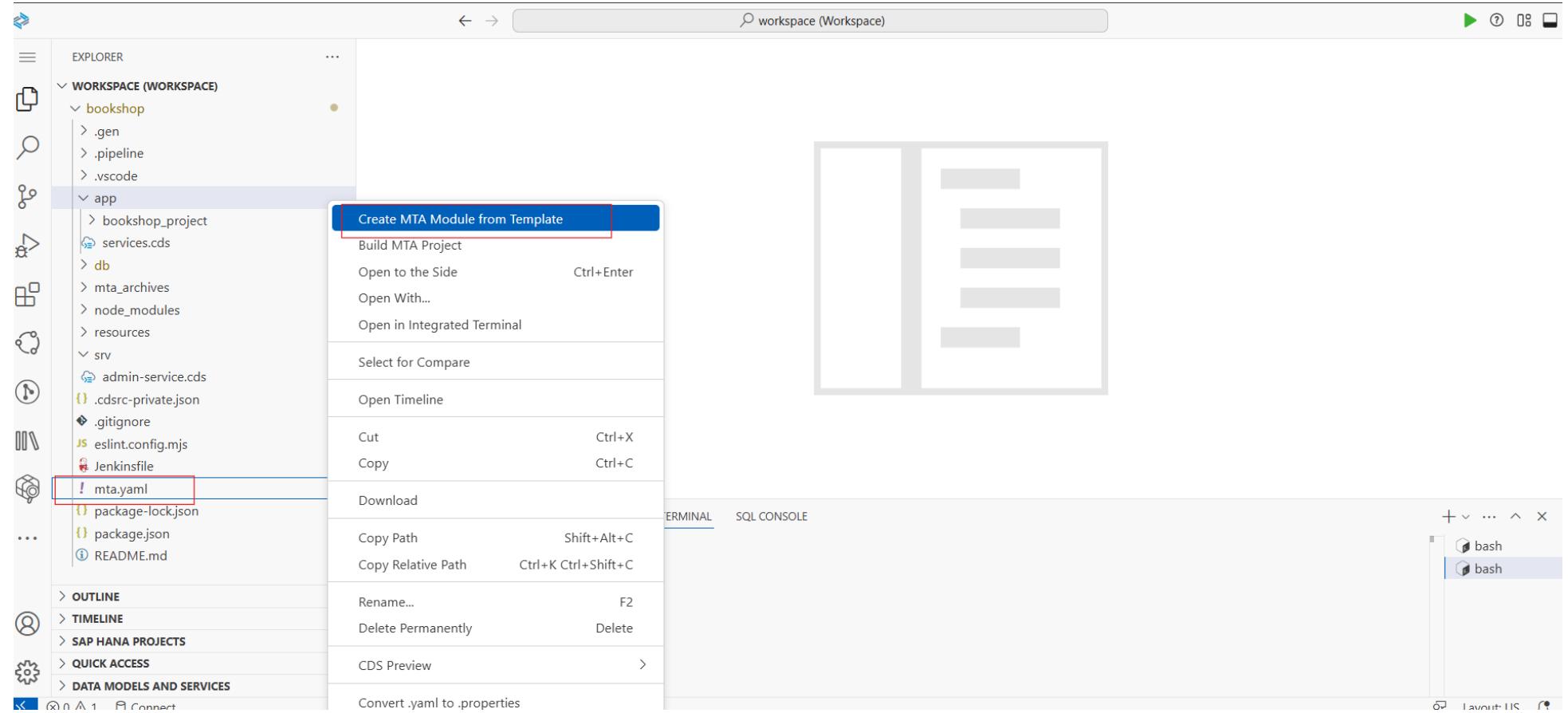
admin-service.cds

using { sap.capire.bookshop as my } from './db/schema';
service AdminService @(requires:'any'){
    entity Books as projection on my.Books;
    entity Authors as projection on my.Authors;
}

```

5.Create MTA Module from Template

In the **MTA (Multi-Target Application)** file, right-click and select **Create MTA Module from Template**.



6.Create a Freestyle UI5 Application with SAP Fiori Generator

SAP Studio interface showing the 'New MTA Module From Template' dialog.

The Explorer sidebar shows the project structure:

- WORKSPACE (WORKSPACE)
 - bookshop
 - .gen
 - .pipeline
 - .vscode
 - app
 - bookshop_project
 - services.cds
 - db
 - mta_archives
 - node_modules
 - resources
 - srv
 - admin-service.cds
 - .cdsrc-private.json
 - .gitignore
 - eslint.config.mjs
 - Jenkinsfile
 - mta.yaml
 - package-lock.json
 - package.json
 - README.md
- OUTLINE
- TIMELINE
- SAP HANA PROJECTS
- QUICK ACCESS
- DATA MODELS AND SERVICES

New MTA Module From Template

Select Module Template

SAP Fiori generator

Path to the selected project's 'mta.yaml' file: /home/user/projects/bookshop/mta.yaml

Templates ⓘ *

Approuter Configuration

Add an approuter configuration to your multitarget application (MTA) project.

[More Information](#)



SAP HANA Database ...

A database module contains design-time database artifacts. On deployment, a database module is assigned an HDI container to which the corresponding run-time database objects are deployed.

[More Information](#)



SAP Fiori generator

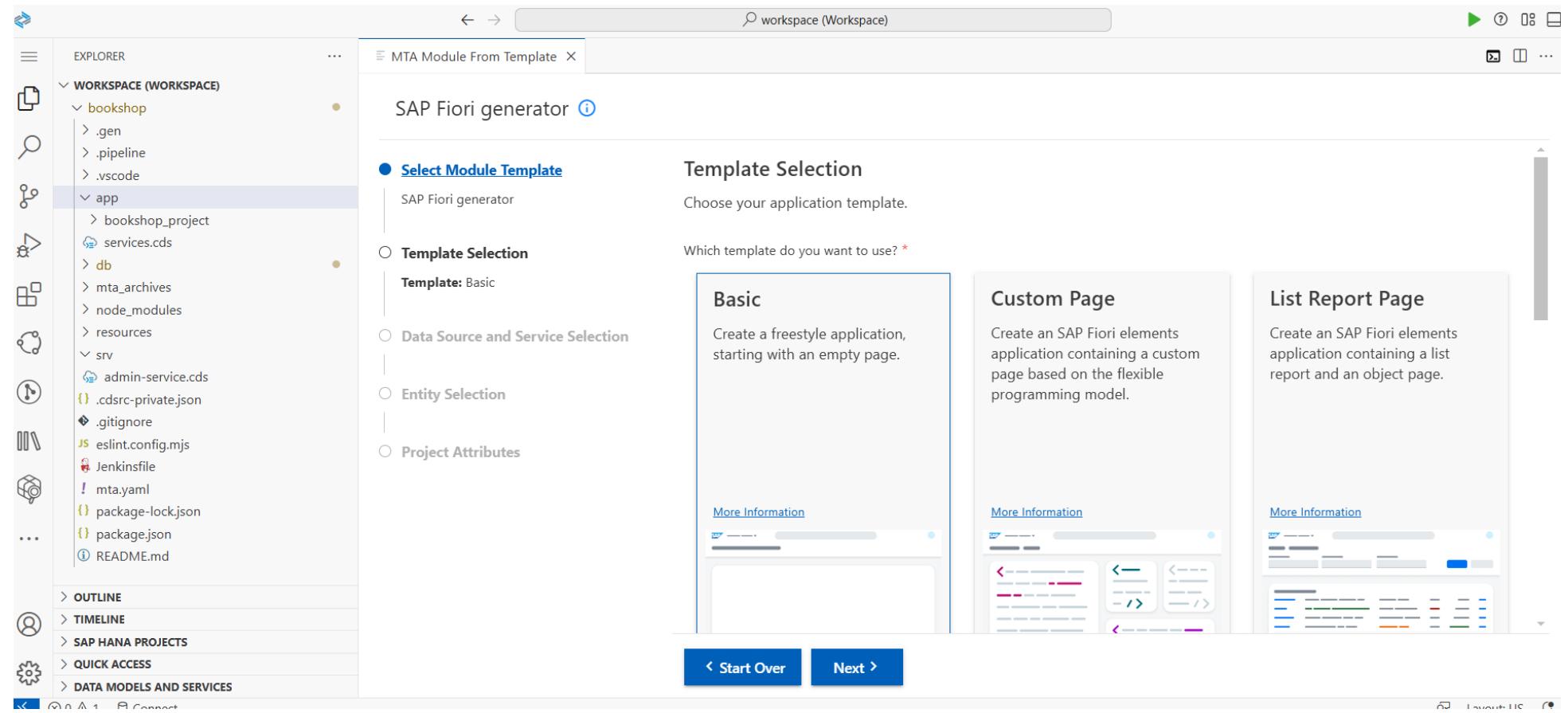
Create an SAPUI5 application using SAP Fiori elements or a freestyle approach

[More Information](#)



Start >

7. Select Basic and Next



The screenshot shows the SAP Fiori generator interface within a workspace. The left sidebar displays the project structure under 'WORKSPACE (WORKSPACE)'. The 'app' folder is selected, showing files like .gen, .pipeline, .vscode, bookshop_project, services.cds, db, mta_archives, node_modules, resources, and srv. Subfolders include admin-service.cds, .cdsrc-private.json, .gitignore, eslint.config.mjs, Jenkinsfile, mta.yaml, package-lock.json, package.json, and README.md.

The main area is titled 'Template Selection' with the sub-section 'Select Module Template' currently active. It asks 'Choose your application template.' and 'Which template do you want to use? *'. Three options are shown:

- Basic**: Create a freestyle application, starting with an empty page. This option is highlighted with a blue border.
- Custom Page**: Create an SAP Fiori elements application containing a custom page based on the flexible programming model.
- List Report Page**: Create an SAP Fiori elements application containing a list report and an object page.

At the bottom, there are 'More Information' links for each template and buttons for 'Start Over' and 'Next >'.

8. Select Local CAP Project and Service

1. Choose the **service** (such as admin-service.cds) that you defined within the CAP project.

The screenshot shows the SAP Fiori generator interface for creating an MTA module from a template. On the left, the Explorer view displays a workspace named 'bookshop' containing an 'app' folder. The 'app' folder is selected, revealing its contents: '.gen', '.pipeline', '.vscode', 'bookshop_project', 'services.cds', 'db', 'mta_archives', 'node_modules', 'resources', 'srv', 'admin-service.cds', '.cdsrc-private.json', '.gitignore', 'eslint.config.mjs', 'Jenkinsfile', 'mta.yaml', 'package-lock.json', 'package.json', and 'README.md'. Below these are links for 'OUTLINE', 'TIMELINE', 'SAP HANA PROJECTS', 'QUICK ACCESS', and 'DATA MODELS AND SERVICES'. At the bottom of the Explorer view, there are navigation icons for back, forward, and search, along with a 'Connect' button. The main workspace area has a title bar 'MTA Module From Template' and a search bar 'workspace (Workspace)'. The central content area is titled 'SAP Fiori generator' and contains several sections:

- Select Module Template:** Shows 'SAP Fiori generator' as the selected template.
- Template Selection:** Shows 'Template: Basic'.
- Data Source and Service Selection:** A section titled 'Configure the data source and select a service.' It includes a dropdown for 'Data source *' set to 'Use a Local CAP Project', a dropdown for 'Choose your CAP project *' set to 'bookshop', and a dropdown for 'OData service *' set to 'AdminService (Node.js)'.
- Entity Selection:** A section for selecting entities.
- Project Attributes:** A section for setting project attributes.

At the bottom of the workspace area are 'Back' and 'Next' buttons. The status bar at the bottom right shows 'Layout: US' and other system information.

9. Enter Project Attributes

The screenshot shows the SAP Fiori generator interface within a workspace. The left sidebar displays the project structure under 'WORKSPACE (WORKSPACE)'. The main area is titled 'MTA Module From Template' and shows the 'Project Attributes' configuration screen.

Select Module Template: SAP Fiori generator

Template Selection: Basic

Data Source and Service Selection:

- Data Source: Use a Local CAP Project
- CAP Project: bookshop
- OData Service: AdminService (Node.js)

Entity Selection:

- View Name: HomePage

Project Attributes:

- Module Name: retail_bookstore
- Application Title: Retail BookStore Project
- Application Namespace: com.bookstore
- Description: An SAP Fiori application.

Deployment Configuration:

- Add deployment configuration to MTA project (/home/user/projects/bookshop/mta.yaml)
 Yes No
- Add PLP configuration
 Yes No
- Configure advanced options
 Yes No

Buttons: < Back, Next >

The generated project will not open in a stand-alone folder.

10. Select Cloud Foundry (earlier which is used in CAP creation)

The screenshot shows the SAP Fiori generator interface within an SAP Studio workspace. The left sidebar displays the project structure under 'WORKSPACE (WORKSPACE)'. The 'app' folder is selected, revealing subfolders like 'bookshop_project', 'services.cds', 'db', 'mta_archives', 'node_modules', 'resources', 'srv', 'admin-service.cds', '.cdsrc-private.json', '.gitignore', 'eslint.config.mjs', 'Jenkinsfile', 'mta.yaml', 'package-lock.json', 'package.json', and 'README.md'. The main panel is titled 'MTA Module From Template' and contains several configuration sections:

- Select Module Template**: SAP Fiori generator
- Deployment Configuration**: Configure deployment settings. The 'Template' dropdown is set to 'Basic'. The 'Please choose the target' dropdown is set to 'Cloud Foundry'. The 'Destination name' dropdown is set to 'Local CAP Project API (Instance Based Destination)'.
- Template Selection**: Basic
- Data Source and Service Selection**: Data Source: Use a Local CAP Project; CAP Project: bookshop; OData Service: AdminService (Node.js)
- Entity Selection**: View Name: HomePage
- Project Attributes**: Module Name: retail_bookstore; Application Title: Retail BookStore Project; Application Namespace: com.bookstore; Description: An SAP Fiori application. A 'More...' link is present.
- Deployment Configuration**: Target Type: Cloud Foundry; Destination Name: Local CAP Project API (Instance Based Destination). Buttons for 'Back' and 'Finish' are visible.

A note at the bottom states: "The generated project will not open in a stand-alone folder."

11. Application is Ready

The screenshot shows the SAP Cloud Platform Application Management interface. On the left is the Explorer sidebar with project navigation. The main area displays the 'Application Information' page for a project named 'Retail BookStore Project'.

Project Details

Identifier	com.bookstore.retailbookstore	Type	SAPUI5 freestyle
Title	Retail BookStore Project	Min UI5 Version	1.132.1
Namespace	com.bookstore	CAP Project Type	CAP Node.js
Location	/home/user/projects/bookshop/app...	Main Service	admin (V4.0)
Files	package.json manifest.json		

Status

What you can do:

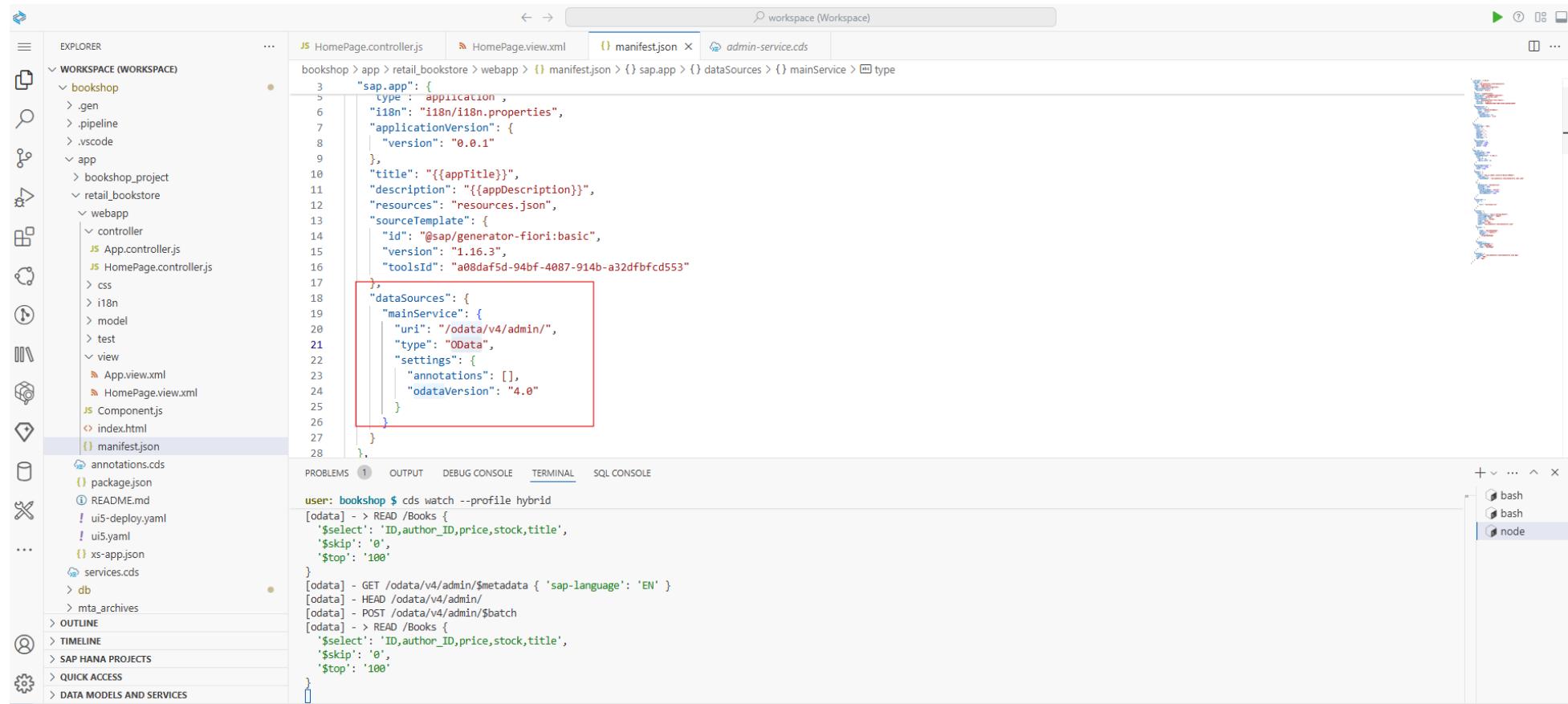
- Preview Application: Choose from start scripts to run the application preview.
- Open Page Map: Open the page map that shows application pages and navigation paths.
- Add Deploy Config: Add deploy configuration.
- Add Fiori Launchpad Config: Add Flp configuration to the application.
- Deploy: Deploy according to the configuration by default stored in 'ui5-deploy.yaml'.
- Undeploy: Remove a deployed artifact from backend or cloud.
- Check Node Modules: Execute command that checks all node module dependencies for newer versions.
- Create Archive: Zip the project excluding the node_modules for sharing (e.g. support cases).
- Change minUI5Version: Change the minimum version of
- Run UI5 Linter: Run a static code analysis to
- Convert Preview Config: Convert the configuration to

The message "The files have been generated." is displayed at the bottom right.

12.Adding the dataSources in manifest.json:

Adding the dataSources section in manifest.json binds your SAP UI5 application to the OData service exposed by your CAP project. It defines the service URL, specifies the OData version (4.0), and sets up the necessary configurations for communication between frontend and backend. This is crucial for making CRUD operations and retrieving data.

```
"dataSources": { "mainService": { "uri": "/odata/v4/admin/", "type": "OData", "settings": { "annotations": [], "odataVersion": "4.0" } } }
```



```
3   "sap.app": {
4     "type": "application",
5     "i18n": "i18n/i18n.properties",
6     "applicationVersion": {
7       "version": "0.0.1"
8     },
9     "title": "{{appTitle}}",
10    "description": "{{appDescription}}",
11    "resources": "resources.json",
12    "sourceTemplate": {
13      "id": "@sap/generator-fiori:basic",
14      "version": "1.16.3",
15      "toolsId": "a08daf5d-94bf-4087-914b-a32dfbfcd553"
16    },
17  },
18  "dataSources": {
19    "mainService": {
20      "uri": "/odata/v4/admin/",
21      "type": "OData",
22      "settings": {
23        "annotations": [],
24        "odataVersion": "4.0"
25      }
26    }
27  },
28}
```

The screenshot shows the SAP Studio IDE interface with the manifest.json file open in the central editor area. The code editor has syntax highlighting for JSON. A red rectangular box highlights the 'dataSources' section, specifically the 'mainService' configuration. The 'manifest.json' tab is selected in the top navigation bar. To the left is the Explorer view showing the project structure, and to the right is the Problems view showing a command-line interface output for a 'cds watch' command. The bottom right corner shows a terminal window with three tabs: bash, bash, and node.

13. Screen Shots and code for xml view and controller

The screenshot shows the SAP Studio IDE interface with the following details:

- EXPLORER View:** Shows the project structure under "WORKSPACE (WORKSPACE)". Key files listed include: .gen, .pipeline, .vscode, app, bookshop_project, retail_bookstore, webapp, controller, App.controller.js, HomePage.controller.js, css, i18n, model, test, view, App.view.xml, HomePage.view.xml, Component.js, index.html, manifest.json, annotations.cds, package.json, README.md, ui5-deploy.yaml, ui5.yaml, xs-app.json, services.cds, db, mta_archives, OUTLINE, SAP HANA PROJECTS, QUICK ACCESS, and DATA MODELS AND SERVICES.
- EDITOR View:** The active file is HomePage.view.xml. The XML code defines a table with four columns: ID, Title, Author ID, Price, and Stock. The table has a path of '/Books'.
- TERMINAL View:** Shows the output of a command run in the terminal window:

```
user: bookshop $ cds watch --profile hybrid
$top': '100'
}
[odata] - GET /odata/v4/admin/$metadata { 'sap-language': 'EN' }
[odata] - HEAD /odata/v4/admin/
[odata] - POST /odata/v4/admin/$batch
[odata] - > READ /Books {
  '$select': 'ID,author_ID,price,stock,title',
  '$skip': '0',
  '$top': '100'
}

[cds] - my watch has ended.
user: bookshop $
```

To connect the OData service to your UI5 application, you can use the following code:

```
var oBooksModel = new ODataModel({
  serviceUrl: "/odata/v4/admin/" // The service URL should point to the root, not to an entity set
});

// Set the OData model to the view
this.getView().setModel(oBooksModel, "BooksModel");
```

This code creates a new **ODataModel** with the service URL pointing to the root of the OData service and binds it to the view with the model name BooksModel. This allows your UI to interact with the data from the backend OData service.

The screenshot shows the SAP Studio IDE interface. On the left is the Explorer view, which lists the project structure under 'WORKSPACE (WORKSPACE)'. The 'controller' folder contains 'App.controller.js' and 'HomePage.controller.js'. The 'view' folder contains 'App.view.xml' and 'HomePage.view.xml'. The 'manifest.json' file is also visible. The central part of the screen is the code editor, showing the provided JavaScript code for creating an OData model and setting it to the view. Below the code editor is the terminal window, which displays the output of the 'cds watch --profile hybrid' command. The terminal shows various CDS metadata requests being processed, such as reading books with specific select, skip, and top parameters, and reading the metadata for the /odata/v4/admin endpoint. On the right side of the interface, there are several toolbars and a status bar at the bottom.

```
var oBooksModel = new ODataModel({
  serviceUrl: "/odata/v4/admin/" // The service URL should point to the root, not to an entity set
};

// Set the OData model to the view
this.getView().setModel(oBooksModel, "BooksModel");
```

```
user: bookshop $ cds watch --profile hybrid
[odata] -> READ /Books {
  '$select': 'ID,author_ID,price,stock,title',
  '$skip': '0',
  '$top': '100'
}
[odata] - GET /odata/v4/admin/$metadata { 'sap-language': 'EN' }
[odata] - HEAD /odata/v4/admin/
[odata] - POST /odata/v4/admin/$batch
[odata] -> READ /Books {
  '$select': 'ID,author_ID,price,stock,title',
  '$skip': '0',
  '$top': '100'
}
```

View.xml

```
<mvc:View controllerName="com.bookstore.retailbookstore.controller.HomePage"
  xmlns:mvc="sap.ui.core.mvc"
  xmlns="sap.m">
  <Page id="page" title="{i18n>title}">

    <Table
      id="booksTable"
      items="{
        path: '/Books'
      }">
      <columns>
        <Column id="_IDGenColumn">
          <Text id="_IDGenText1" text="Title" />
        </Column>
        <Column id="_IDGenColumn1">
          <Text id="_IDGenText2" text="Author ID" />
        </Column>
        <Column id="_IDGenColumn2">
          <Text id="_IDGenText3" text="Price" />
        </Column>
        <Column id="_IDGenColumn3">
          <Text id="_IDGenText4" text="Stock" />
        </Column>
      </columns>

      <items>
        <ColumnListItem id="_IDGenColumnListItem">
          <cells>
            <Text id="_IDGenText5" text="{title}" />
          </cells>
        </ColumnListItem>
      </items>
    </Table>
  </Page>
</mvc:View>
```

```
<Text id="_IDGenText6" text="{author_ID}" />
<Text id="_IDGenText7" text="{price}" />
<Text id="_IDGenText8" text="{stock}" />
</cells>
</ColumnListItem>
</items>
</Table>
</Page>
</mvc:View>
```

Controller.js

```
sap.ui.define([
  "sap/ui/core/mvc/Controller",
  "sap/ui/model/odata/v4/ODataModel"
], (Controller,ODataModel) => {
  "use strict";

  return Controller.extend("com.bookstore.retailbookstore.controller.HomePage", {
    onInit() {

      var oBooksModel = new ODataModel({
        serviceUrl: "/odata/v4/admin/" // The service URL should point to the root, not to an entity set
      });

      // Set the OData model to the view
      this.getView().setModel(oBooksModel, "BooksModel");

    }
  });
});
```

14. Application Ready – Click on Retail Bookstore

Once the application is fully set up and deployed, click on **Retail Bookstore** to access and interact with the application. This will launch the UI5 interface connected to your backend OData service.

Welcome to @sap/cds Server

Serving bookshop 1.0.0

These are the paths currently served:

Web Applications:

/bookshop_project/dist
/bookshop_project/webapp
/retail_bookstore/webapp

Service Endpoints:

/odata/v4/admin / \$metadata		
Books		Fiori preview
Authors		Fiori preview
Genres		Fiori preview
Currencies		Fiori preview

This is an automatically generated page.
You can replace it with a custom `./app/index.html`.

15.Application Final: Table View with All Data

Once the application is ready, the **Retail Bookstore** will display a table view populated with all the data from the backend OData service. This table will show a list of items, such as books, retrieved from the database, providing users with an interactive and dynamic view of the data.

Retail BookStore Project

Title	Author ID	Price	Stock
The Art of Coding	101	29.99	50
Database Essentials	102	19.99	30
Cloud Computing	103	39.99	20

Thank you for reading **Chapter 1: Implementing CRUD Operations – Read**.
The next chapter, **Create**, will be available soon. Stay tuned for more!

Thank You