In SAP Fiori, the Model-View-Controller (MVC) concept is a design pattern used to separate the application's logic into three interconnected components:

1. Model:

The Model represents the data layer. It contains the application's business logic, data retrieval, and data handling mechanisms. The Model interacts with the backend systems (like OData services in SAP Fiori) to fetch, manipulate, and update the data displayed in the UI. Models in SAP Fiori can be JSON, XML, or OData models.

* SAPUI5 supports different types of models:

1. JSON Model: For storing and handling data in JSON format. Ideal for frontend logic and local data handling.
2. XML Model: For XML-based data.
3. OData Model: For retrieving and sending data from/to OData services. Most often used in enterprise scenarios where backend integration is required (e.g., SAP services).
4. Resource Model: For storing language-specific text in i18n files, enabling internationalization.

Key responsibilities:

* Manages the data and business logic.
* Handles communication with backend services.
* Provides data to the View through binding.

1. View:

The View represents the user interface (UI) of the application. It defines the structure and layout of the screens that the user interacts with. In SAP Fiori, the View is often written in XML format and focuses on how the data (provided by the Model) is presented.

Key responsibilities:

* Defines how data is displayed to the user.
* Does not contain any business logic.
* Binds UI elements to the Model's data.

1. Controller:

The Controller is the intermediary between the Model and the View. It handles user inputs (like button clicks or form submissions) and processes them to update the Model or manipulate the View. Controllers are written in JavaScript in SAP Fiori applications.

Key responsibilities:

* Responds to user actions and events.
* Updates the Model and View accordingly.
* Implements the application's logic, including input validation, event handling, and data flow management.

How MVC Works in SAP Fiori:

* The View is responsible for displaying the data from the Model, but it doesn't handle the data directly. It uses data binding to connect to the Model and present the information in UI elements (like tables, input fields, etc.).
* The Controller listens for user actions (e.g., button clicks or form submissions) and processes those actions. It might call the backend service through the Model to retrieve new data or update existing data.
* The Model communicates with the backend to retrieve or send data. When the Model’s data changes, the View is automatically updated if data binding is used.

Example:

In a typical SAP Fiori application:

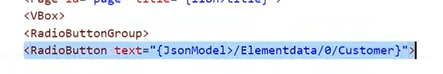
* Model: An OData model retrieves product details from a backend system (e.g., product name, price, and stock).
* View: Displays this data in a table format to the user.
* Controller: Handles user actions like sorting the table or filtering products. If the user selects a product, the controller might fetch additional details from the Model and display them.

This separation of concerns allows developers to manage, extend, and maintain the application more efficiently, as changes in one component (Model, View, or Controller) do not directly affect the others.

JSON:

In model create json file and goto manifest.jsn add your data source. Goto view and mapped data with json file.

Once the json file is ready (in model) then assign this data source in manifest.jsn ( if flex enable is true then make it as false ). Add your json file in models. ( already exists i18n model is there after that u can add your model ). Give type(“sap.ui.model.json.json.model”) and uri(“model/mockdata.json”).

Goto view use above syntax 

Here Jsonmodel (is the name given in manifest.jsn)

Elementdata ( is the name of the array which is given model json file)

1. ( is the array number)

Customer (is the array field name)

You can check in <https://jsonlint.com>

List report:

we can use the List Report as replacement of ALV in SAP GUI.

SAP UI5 Libraries:

Controller libraries are:



Core libraries are:

