**1. Creare il Fragment**

Crea un file chiamato **HelloDialog.fragment.xml** e inserisci il seguente codice:

<mvc:View controllerName="project2.controller.View1"

    xmlns:mvc="sap.ui.core.mvc" displayBlock="true"

    xmlns="sap.m">

    <Page id="page" title="{i18n>title}">

      <Button id="buttonOpenDialog" text="Apri il Dialog" press=".onOpenDialog"></Button>

    </Page>

</mvc:View>

**2. Creare il Controller del Fragment**

Crea un file chiamato **HelloDialog.js** e inserisci il seguente codice:  
sap.ui.define([

    "sap/ui/base/ManagedObject",

    "sap/ui/core/Fragment"

  ], function(ManagedObject, Fragment) {

    return ManagedObject.extend("project2.view.Fragments.HelloDialog", {

      constructor: function(oView) {

        this.\_oView = oView;

      },

      exit: function() {

        delete this.\_oView;

      },

      open: function() {

        var oView = this.\_oView;

        if (!oView.byId("HelloDialog")) {

          var oFragmentController = {

            onCloseDialog: function() {

              oView.byId("HelloDialog").close();

            }

          };

          Fragment.load({

            id: oView.getId(),

            name: "project2.view.Fragments.HelloDialog",

            controller: oFragmentController

          }).then(function(oDialog) {

            oView.addDependent(oDialog);

            oDialog.open();

          });

        } else {

          oView.byId("HelloDialog").open();

        }

      }

    });

  });

**3. Inizializzare nel Componente Principale**

Nel tuo file **Component.js**, aggiungi il seguente codice:

sap.ui.define([

        "sap/ui/core/UIComponent",

        "sap/ui/Device",

        "project2/model/models",

        "project2/controller/Fragments/HelloDialog"

    ],

 function (UIComponent, Device, models , HelloDialog)

init: function () {

                // call the base component's init function

                UIComponent.prototype.init.apply(this, arguments);

                // enable routing

                this.getRouter().initialize();

                // set the device model

                this.setModel(models.createDeviceModel(), "device");

                this.\_HelloDialog = new HelloDialog(this.getRootControl());

            },

            exit: function() {

                this.\_HelloDialog.destroy();

                delete this.\_HelloDialog;

              },

              openHelloDialog: function() {

                this.\_HelloDialog.open();

              }

Cosa Succede qui?

HelloDialog(this.getRootControl()); -- Passo al costruttore del controller di HelloDialog il riferimento alla View radice

In che contesti viene invocato il metodo Exit?

1. **Navigazione**: Se l'utente naviga fuori dalla vista corrente o dal componente, il framework potrebbe decidere di distruggere il componente per liberare risorse.
2. **Chiusura dell'Applicazione**: Se l'utente chiude l'applicazione, tutti i componenti e i loro rispettivi oggetti e risorse saranno distrutti, e il metodo **exit** verrà chiamato.
3. **Ricarica della Pagina**: Se l'utente ricarica la pagina web, il componente verrà distrutto e ricreato, e il metodo **exit** verrà chiamato durante la fase di distruzione.
4. **Cambio di Route**: Se la tua applicazione utilizza un sistema di routing e l'utente cambia la route, il componente corrente potrebbe essere distrutto, facendo scattare il metodo **exit**.

**4. Utilizzare in Vari Controller**

In ogni controller in cui desideri utilizzare il dialogo, puoi chiamare il metodo **openMyDialog** del componente principale:

onOpenDialog: function() {

                this.getOwnerComponent().openHelloDialog();

              }

**5. Pulizia delle Risorse**

La pulizia delle risorse è già gestita nei metodi **exit** dei vari componenti e controller.

Con questi passaggi, dovresti essere in grado di creare un dialogo riutilizzabile in SAP UI5.