## We have looked at View and Controller lets bring in Model.

Lets add description field to each of the input fields. Now whenever we type Invoice number or type or anything it shall update the description automatically via binding.
Change App view to

```
<mvc:View
   controllerName="search.Invoice.controller.App"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc">
   <Input placeholder="Invoice Type" value="{/invoiceInput/invType}" description="Invoice Type is {/invoiceInput/invType} "
valueLiveUpdate="true"/>
   <Input placeholder="Invoice Number" value="{/invoiceInput/invNumber}" description="Entered Invoice Number is
{/invoiceInput/invNumber}" valueLiveUpdate="true"/>
   <Input placeholder="Company Code" value="{/invoiceInput/invCompanyCode}" description="Company Code is
{/invoiceInput/invCompanyCode} " valueLiveUpdate="true"/>
      <Button
      text="Search"
      press=".onSearchInvoices"/>
</mvc:View>
```

Change App controller to

```
sap.ui.define([
   "sap/ui/core/mvc/Controller",
   "sap/m/MessageToast",
   "sap/ui/model/json/JSONModel"
], function (Controller,MessageToast,JSONModel) {
   "use strict";
   return Controller.extend("search.Invoice.controller.App", {
      onInit: function(){
         // Set data model on view.
         var invInput = {
            invInput : {
               invNumber : "",
               invType:"",
               invCompanyCode:""
            }
         };
         var oModel = new JSONModel(invInput);
         this.getView().setModel(oModel);
      },
      onSearchInvoices : function () {

         MessageToast.show("Searching Invoice's");
      }
   });
});
```

## Lets now think about all the labels which we have

Different countries have different languages how do we handle via i18n
Create the folder webapp/i18n and the file i18n.properties inside
Add the following to i18n

```
searchInvoice=Searching Invoice's
invNumber=Invoice Number
invType=Invoice Type
invCompanyCode=Company Code
```

Change the App.controller.js to

```
sap.ui.define([
   "sap/ui/core/mvc/Controller",
   "sap/m/MessageToast",
   "sap/ui/model/json/JSONModel",
     "sap/ui/model/resource/ResourceModel"
], function (Controller,MessageToast,JSONModel,ResourceModel) {
   "use strict";
   return Controller.extend("search.Invoice.controller.App", {
      onInit: function(){
         // Set data model on view.
         var invInput = {
            invInput : {
               invNumber : "",
               invType:"",
               invCompanyCode:""
            }
         };
         var oModel = new JSONModel(invInput);
         this.getView().setModel(oModel);

         var i18nModel = new ResourceModel({
            bundleName: "search.Invoice.i18n.i18n"
         });
         this.getView().setModel(i18nModel, "i18n");
      },
      onSearchInvoices : function () {
         var oBundle = this.getView().getModel("i18n").getResourceBundle();
         MessageToast.show(oBundle.getText("searchInvoice"));
      }
   });
});
```

Change the App.view.xml

```
<mvc:View
   controllerName="search.Invoice.controller.App"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc"
   >
   <Input placeholder="{i18n>invNumber}" value="{/invoiceInput/invType}" description="{i18n>invNumber} is {/invoiceInput/invType} "
valueLiveUpdate="true"/>
   <Input placeholder="{i18n>invType}" value="{/invoiceInput/invNumber}" description="{i18n>invType} is {/invoiceInput/invNumber}"
valueLiveUpdate="true"/>
   <Input placeholder="{i18n>companyCode}" value="{/invInput/companyCode}" description="{i18n>companyCode}  is
{/invInput/companyCode}" valueLiveUpdate="true"/>
      <Button
      text="Search"
      press="onSearchInvoices"/>
</mvc:View>
```

## Lets bring in Component concept.

Lets think big if we have to embed our app in SAP Fiori Lanuchpad or it needs be opened in the flp container then we have to make it independent of index.html it has to be something like a component which anyone can use.

Create Component.js file in webapp folder.

```
sap.ui.define([
   "sap/ui/core/UIComponent",
    "sap/ui/model/json/JSONModel",
   "sap/ui/model/resource/ResourceModel"
], function (UIComponent,JSONModel,ResourceModel) {
   "use strict";
   return UIComponent.extend("search.Invoice.Component", {
 metadata : {
         rootView: {
            "viewName": "search.Invoice.view.App",
            "type": "XML",
            "async": true,
            "id": "app"
         }
      },
      init : function () {
         // call the init function of the parent
         UIComponent.prototype.init.apply(this, arguments);
          // Set data model on view.
         var invInput = {
            invInput : {
               invNumber : "",
               invType:"",
               invCompanyCode:""
            }
         };
         var oModel = new JSONModel(invInput);
         this.setModel(oModel);

         var i18nModel = new ResourceModel({
            bundleName: "search.Invoice.i18n.i18n"
         });
         this.setModel(i18nModel, "i18n");
      }
   });
});
```

Change App.controller.js. Remove the model from app controller as we have defined in the Component controller. The childrens have access to parent so they will be able to access model.

```
sap.ui.define([
   "sap/ui/core/mvc/Controller",
   "sap/m/MessageToast"
], function (Controller,MessageToast) {
   "use strict";
   return Controller.extend("search.Invoice.controller.App", {
      onInit: function(){
      },
      onSearchInvoices : function () {
         var oBundle = this.getView().getModel("i18n").getResourceBundle();
         MessageToast.show(oBundle.getText("searchInvoice"));
      }
   });
});
```

Change Index.js to now create the component container

```
sap.ui.define([
"sap/ui/core/ComponentContainer"
], function (ComponentContainer) {
    "use strict";


    new ComponentContainer({
        name: "search.Invoice",
        settings : {
            id : "walkthrough"
        },
        async: true
    }).placeAt("content");

});
```

## Bring in next biggie Descriptor manifest.json

We have got rid of Index.html dependency. Our view and models are associated via component.js. But component.js is handling
multiple things. It makes sense to move all config out of it into some kind of description which describe the application and component takes care of application coding parts.

Lets create file manifest.json in webapp folder.

```
{
  "_version": "1.12.0",
  "sap.app": {
    "id": "search.Invoice",
    "type": "application",
    "i18n": "i18n/i18n.properties",
    "title": "{{appTitle}}",
    "description": "{{appDescription}}",
    "applicationVersion": {
      "version": "1.0.0"
    }
  },
  "sap.ui": {
    "technology": "UI5",
    "deviceTypes": {
        "desktop": true,
        "tablet": true,
        "phone": true
    }
  },
  "sap.ui5": {
    "rootView": {
        "viewName": "search.Invoice.view.App",
        "type": "XML",
        "async": true,
        "id": "app"
    },
    "dependencies": {
      "minUI5Version": "1.60",
      "libs": {
        "sap.m": {}
      }
    },
    "models": {
      "i18n": {
        "type": "sap.ui.model.resource.ResourceModel",
        "settings": {
          "bundleName": "search.Invoice.i18n.i18n"
        }
      }
    }
  }
}
```

Modify the index.html file to call our component directly without using index.js

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Purchase Order Search App</title>
        <script
        id="sap-ui-bootstrap"
        src="https://openui5.hana.ondemand.com/resources/sap-ui-core.js"
        data-sap-ui-theme="sap_belize"
        data-sap-ui-libs="sap.m"
        data-sap-ui-compatVersion="edge"
        data-sap-ui-async="true"
        data-sap-ui-oninit="module:sap/ui/core/ComponentSupport"
        data-sap-ui-resourceroots='{
            "search.Invoice": "./"
        }'>

    </script>
</head>
<body class="sapUiBody" id="content">

<div data-sap-ui-component data-name="search.Invoice" data-id="container" data-settings='{"id" : "walkthrough"}'></div>

</body>
</html>
```

Add app title and description in i18n

```
searchInvoice=Searching Invoice's
invNumber=Invoice Number
invType=Invoice Type
invCompanyCode=Company Code
appTitle=My Title
appDescription=my desc
```

Change component js now to refer manifest.json and remove resource model

```
sap.ui.define([
   "sap/ui/core/UIComponent",
    "sap/ui/model/json/JSONModel",
   "sap/ui/model/resource/ResourceModel"
], function (UIComponent,JSONModel,ResourceModel) {
   "use strict";
   return UIComponent.extend("search.Invoice.Component", {
 metadata : {
            manifest: "json"
      },
      init : function () {
         // call the init function of the parent
         UIComponent.prototype.init.apply(this, arguments);
          // Set data model on view.
         var invInput = {
            invInput : {
               invNumber : "",
               invType:"",
               invCompanyCode:""
            }
         };
         var oModel = new JSONModel(invInput);
         this.setModel(oModel);

      }
   });
});
```

## Our structure is ready Lets beautify it.

Lets move our view inside pages change App.view.xml

```
<mvc:View
   controllerName="search.Invoice.controller.App"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc"
displayBlock="true">

     <App>
      <pages>
         <Page title="{i18n>homePageTitle}">
            <content>
   <Input placeholder="{i18n>invNumber}" value="{/invoiceInput/invType}" description="{i18n>invNumber} is {/invoiceInput/invType} "
valueLiveUpdate="true"/>
   <Input placeholder="{i18n>invType}" value="{/invoiceInput/invNumber}" description="{i18n>invType} is {/invoiceInput/invNumber}"
valueLiveUpdate="true"/>
   <Input placeholder="{i18n>companyCode}" value="{/invInput/companyCode}" description="{i18n>companyCode}  is
{/invInput/companyCode}" valueLiveUpdate="true"/>
      <Button
      text="Search"
      press="onSearchInvoices"/>
       </content>
         </Page>
      </pages>
   </App>

</mvc:View>
```

Lets add our app inside a shell infact

```
<mvc:View
   controllerName="search.Invoice.controller.App"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc"
displayBlock="true">
<Shell>
     <App>
      <pages>
         <Page title="{i18n>homePageTitle}">
            <content>
   <Input placeholder="{i18n>invNumber}" value="{/invoiceInput/invType}" description="{i18n>invNumber} is {/invoiceInput/invType} "
valueLiveUpdate="true"/>
   <Input placeholder="{i18n>invType}" value="{/invoiceInput/invNumber}" description="{i18n>invType} is {/invoiceInput/invNumber}"
valueLiveUpdate="true"/>
   <Input placeholder="{i18n>companyCode}" value="{/invInput/companyCode}" description="{i18n>companyCode}  is
{/invInput/companyCode}" valueLiveUpdate="true"/>
      <Button
      text="Search"
      press="onSearchInvoices"/>
       </content>
         </Page>
      </pages>
   </App>
   </Shell>
</mvc:View>
```

Lets add some margins also..

```
<mvc:View
   controllerName="search.Invoice.controller.App"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc"
displayBlock="true">
<Shell>
     <App>
      <pages>
         <Page title="{i18n>homePageTitle}" >
            <content >
   <Input placeholder="{i18n>invNumber}" value="{/invoiceInput/invType}" description="{i18n>invNumber} is {/invoiceInput/invType} "
valueLiveUpdate="true" class="sapUiResponsiveMargin"
     />
   <Input class="sapUiResponsiveMargin"
     placeholder="{i18n>invType}" value="{/invoiceInput/invNumber}" description="{i18n>invType} is {/invoiceInput/invNumber}"
valueLiveUpdate="true"/>
   <Input  class="sapUiResponsiveMargin"
     placeholder="{i18n>companyCode}" value="{/invInput/companyCode}" description="{i18n>companyCode}  is {/invInput/companyCode}"
valueLiveUpdate="true"/>
      <Button class="sapUiResponsiveMargin"

      text="Search"
      press="onSearchInvoices"/>
       </content>
         </Page>
      </pages>
   </App>
   </Shell>
</mvc:View>
```

## So now time has come to add our list of Invoice's

if we will add list of invoice also in app view it will become too crowded lets have separate view for this complete view and let app view just calls it.

Create a new view Invoices.view.xml and move code

```
<mvc:View
   controllerName="search.Invoice.controller.Invoice"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc">
    <Input placeholder="{i18n>invNumber}" value="{/invoiceInput/invType}" description="{i18n>invNumber} is {/invoiceInput/invType} "
valueLiveUpdate="true" class="sapUiResponsiveMargin"
     />
   <Input class="sapUiResponsiveMargin"
     placeholder="{i18n>invType}" value="{/invoiceInput/invNumber}" description="{i18n>invType} is {/invoiceInput/invNumber}"
valueLiveUpdate="true"/>
   <Input  class="sapUiResponsiveMargin"
     placeholder="{i18n>companyCode}" value="{/invInput/companyCode}" description="{i18n>companyCode}  is {/invInput/companyCode}"
valueLiveUpdate="true"/>
      <Button class="sapUiResponsiveMargin"

      text="Search"
      press="onSearchInvoices"/>
</mvc:View>
```

Move button methods from App.controller.js to Invoice.controller.js, this Invoice controller needs to be created

```
sap.ui.define([
   "sap/ui/core/mvc/Controller",
   "sap/m/MessageToast"
], function (Controller,MessageToast) {
   "use strict";
   return Controller.extend("search.Invoice.controller.Invoice", {
      onInit: function(){
      },
      onSearchInvoices : function () {
         var oBundle = this.getView().getModel("i18n").getResourceBundle();
         MessageToast.show(oBundle.getText("searchInvoice"));
      }
   });
});
```

Add the call of Invoice view in App.view.xml

```
<mvc:View
   controllerName="search.Invoice.controller.App"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc"
displayBlock="true">
<Shell>
     <App>
      <pages>
         <Page title="{i18n>homePageTitle}" >
            <content >
 <mvc:XMLView viewName="search.Invoice.view.Invoice"/>
       </content>
         </Page>
      </pages>
   </App>
   </Shell>
</mvc:View>
```

Remove the methods of app controller

```
sap.ui.define([
   "sap/ui/core/mvc/Controller",
   "sap/m/MessageToast"
], function (Controller,MessageToast) {
   "use strict";
   return Controller.extend("search.Invoice.controller.App", {

   });
});
```

Create a Invoice.json file with all Invoice's

```
{
  "Invoices": [
    {
      "Vbeln": "1",
      "Type":"NB",
      "Bukrs":1000,
      "Amount": 100,
      "InvDate": "2020-06-23T00:00:00",
      "InvStatus": "CONF",
      "Waers":"INR"
    },
    {
      "Vbeln": "2",
      "Type":"NB",
      "Bukrs":2000,
      "Amount": 200,
      "InvDate": "2020-06-21T00:00:00",
      "InvStatus": "CONF",
      "Waers":"INR"
    },
        {
      "Vbeln": "3",
      "Type":"NB",
      "Bukrs":3000,
      "Amount": 300,
      "InvDate": "2020-06-13T00:00:00",
      "InvStatus": "OPEN",
      "Waers":"INR"
    },
        {
      "Vbeln": "4",
      "Type":"NB",
      "Bukrs":4000,
      "Amount": 400,
      "InvDate": "2020-06-12T00:00:00",
      "InvStatus": "PEND",
      "Waers":"INR"
    }
  ]
}
```

Add the call for model in manifest.json

```
{
  "_version": "1.12.0",
  "sap.app": {
    "id": "search.Invoice",
    "type": "application",
    "i18n": "i18n/i18n.properties",
    "title": "{{appTitle}}",
    "description": "{{appDescription}}",
    "applicationVersion": {
      "version": "1.0.0"
    }
  },
  "sap.ui": {
    "technology": "UI5",
    "deviceTypes": {
        "desktop": true,
        "tablet": true,
        "phone": true
    }
  },
  "sap.ui5": {
    "rootView": {
        "viewName": "search.Invoice.view.App",
        "type": "XML",
        "async": true,
        "id": "app"
    },
    "dependencies": {
      "minUI5Version": "1.60",
      "libs": {
        "sap.m": {}
      }
    },
    "models": {
      "i18n": {
        "type": "sap.ui.model.resource.ResourceModel",
        "settings": {
          "bundleName": "search.Invoice.i18n.i18n"
        }
      },
      "inv":{
        "type": "sap.ui.model.json.JSONModel",
        "uri": "Invoice.json"
      }
    }
  }
}
```

Add the list for showing the value in Invoice view

```
<mvc:View
   controllerName="search.Invoice.controller.Invoice"
   xmlns="sap.m"
   xmlns:mvc="sap.ui.core.mvc">
    <Input placeholder="{i18n>invNumber}" value="{/invoiceInput/invType}" description="{i18n>invNumber} is {/invoiceInput/invType} "
valueLiveUpdate="true" class="sapUiResponsiveMargin"
     />
   <Input class="sapUiResponsiveMargin"
     placeholder="{i18n>invType}" value="{/invoiceInput/invNumber}" description="{i18n>invType} is {/invoiceInput/invNumber}"
valueLiveUpdate="true"/>

   <Input  class="sapUiResponsiveMargin"
     placeholder="{i18n>companyCode}" value="{/invInput/companyCode}" description="{i18n>companyCode}  is {/invInput/companyCode}"
valueLiveUpdate="true"/>
      <Button class="sapUiResponsiveMargin"

      text="Search"
      press="onSearchInvoices"/>
      <List
      headerText="{i18n>InvoiceList}"
      class="sapUiResponsiveMargin"
      width="auto"
      items="{inv>/Invoices}" >
      <items>
         <ObjectListItem
            title="{inv>Vbeln} - {inv>Bukrs}"
            number="{
            parts: [{path: 'inv>Amount'}, {path: 'inv>/Waers'}],
            type: 'sap.ui.model.type.Currency',
            formatOptions: {
                showMeasure: false
            }
        }"
        numberUnit="{inv>/Waers}"/>
      </items>
   </List>
</mvc:View>
```