sap.m.List 및 MessageBox 및 filter 및 Combox

sap.m.List 및 MessageBox 및 filter 및 Combox

gpt 로 만들고 내용을 테스트 해보고 수정했음

`sap.m.List`

- *`sap.m.List`*는 UI5에서 리스트 형태로 데이터를 표시할 때 사용하는 컴포넌트입니다. 이 컴포넌트는 각 항목을 **`ListItem`*으로 정의하여 반복적으로 데이터를 표시합니다. 이 리스트는 **정렬** 및 **선택** 기능을 지원하며, **콤보박스**와 **검색창**을 통한 **필터링** 기능도 제공합니다.

기능

- **정렬**: 리스트 항목을 오름차순 또는 내림차순으로 정렬할 수 있습니다.
- **선택 가능**: 사용자가 리스트 항목을 선택할 수 있게 하고, 이를 통해 다른 작업을 트리거할 수 있습니다.
- **필터**: 콤보박스나 검색창을 통해 리스트의 항목을 필터링할 수 있습니다.

main.view.xml

```
<mvc:View xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m"
controllerName="sync.c15.routing.controller.Main">
  <!-- 콤보박스 필터 -->
  <ComboBox id="idComboBox" selectionChange="onComboBoxChange">
  //selectionChange 콤보스 선택 이벤트 (onComboBoxChange 가 작동)
    <items> // 이곳에 입력한 갯수만 큼 콤보 박스에 나온다.
      <core:Item key="1" text="Option 1"/>
      <core:Item key="2" text="Option 2"/>
      <core:Item key="3" text="Option 2"/>
    </items>
```

```
  </ComboBox>

  <!-- 검색창 필터 -->
  <SearchField id="idSearchField" search="onSearch"/> // search 검색입력 (or

  <!-- List (정렬 및 선택 가능) -->
  </List id="idList"
  items="{/items}"
   sorter : [
            {path : '정렬할 field1',
            group : true},

            {path : '정렬할 field2',
            group : true,
            descending : true}
  >
             // 정렬할 필드를 입력 , 여러개 입력가능
             // group 을 사용하여 db 의 group by 처럼 구현가능

    <items>
     <StandardListItem title="{name}"
     description="{description}"
     type="Active"  // 선택 가능
     press="onItemPress"/> // items 의 갯수만큼 StandardListItem 가 출력됨
    </items>
  </List>
</mvc:View>

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
main.controller.js


sap.ui.define(["sap/ui/core/mvc/Controller", "sap/m/MessageBox",],
(Controller ,MessageBox ) => {
  "use strict";
// sap/m/MessageBox 를 사용해야 메세지 박스 나옴.
```

```
sap.ui.controller("sync.c15.routing.controller.Main", {
 onInit() {

 },

 onComboBoxChange (oEvent) {
  // 콤보박스 선택 시 필터 처리
  // 필터랑 기본적으로 똑같음, 콤보박스로 선택을 한것을 필터 한것이여서
    var oComboBox = oEvnet.getSource(); // 이벤트가 발생한 control 가져옴
    var sKey = oComboBox.getSelectedKey(); // 콤보박스가 선택한 값 을 가져옴.

    var aFilter = [];  // 필터를 저장하기위한 배열
                                  // 필터가 여러개를 적용해야 할 수 있어서 배열로 만듬
    var oFilter = null;

    if (sKey !== "1") {
      oFilter = new sap.ui.model.Filter(
        "name", // 조회할 필드명
        sap.ui.model.FilterOperator.EQ, // 조회 조건 EQ, 등등 db 랑 같음
        sKey // 내가 콤보박스에서 선택한 내용
      );
      aFilter.push(oFilter); // 현재 내가 선택한 filter의 내용을 배열에 추가.
    } // afilter에 ofilter2 이런식으로 push 를 추가로 하면 filter를 여러개 적용 가능하다

    var oTable = this.byId("idList"); // 조회를 적용할 List id 을 가져온다.
    var oBinding = oTable.getBinding("items"); // <List > 에서 <items> 부분을 가
    // sap.ui.table일 경우 "rows" 를 적는다
    // sap.m.talbe은 똑같이 "items"
    oBinding.filter(aFilter); //  items 를 출력할때 aFilter 에 등록된 필터 조건을 적용한

!!!!정렬도 응용가능
!!!!정렬도 응용가능
!!!!정렬도 응용가능
// Column을 내림차순 정렬
// 오름 차순 false, 내림 차순 true
 var oSorter = new sap.ui.model.Sorter("적용할 column", true);

 // 정렬 적용
```

```
    // 정렬도 똑같이 []안에 여러개를 넣으면 정렬기준 다중 적용
    oBinding.sort([oSorter]);


    },

    onSearch (oEvent) {
      var query = oEvent.getParameter("query"); // SearchField 에 입력한 값을 받이
      var oList = this.byId("idList"); // 검색 조건을 적용할 list 의 id 를 가져온다.
      var oBinding = oList.getBinding("items"); // <List> 에서 <items> 부분을 가져온
      var oFilter = new sap.ui.model.Filter("name",  // 조회할 필드명
      sap.ui.model.FilterOperator.Contains,  // 조회 조건 EQ, 등등 db 랑 같음
      query // 내가 검색한 내용
      );
      oBinding.filter(oFilter);  //  items 를 출력할때 oFilter 필터 조건을 적용한다.


    },

    onItemPress (oEvent) {
      var oItem = oEvent.getSource(); // 클릭한 list 의 행에서
      var sTitle = oItem.getTitle(); // title="{name}" 을 가져온다.


      // MessageBox 이거도 시험에 나오니까 참고
      MessageBox.show("You selected " + sTitle);

      MessageBox.success("내용") // 성공 메시지
         MessageBox.warning("내용") // 경고 메시지
         MessageBox.error("내용") // 오류 메시지
         MessageBox.alert("내용") // 일반적인 경고 메시지
    }
  });
```

!!Table 에도 fillter 를 적용 가능하다 this.byId("table id") 부분에 table id 를 적으면 된다.

!!Table 에도 fillter 를 적용 가능하다 this.byId("table id") 부분에 table id 를 적으면 된다.

!!Table 에도 fillter 를 적용 가능하다 this.byId("table id") 부분에 table id 를 적으면 된다.

!!Table 에도 fillter 를 적용 가능하다 this.byId("table id") 부분에 table id 를 적으면 된다.

위의 코드를 참고 getBinding()의 input parm이 좀 다르다

정렬도 응용 가능하다!!! 코드 참고

sap.ui.model.FilterOperator 정리

연산자	설명	예시
Contains	부분 일치 검색	"Apple"이 포함된 값 검색
EQ	정확한 값 일치	"Apple"과 정확히 일치하는 값 검색
StartsWith	특정 값으로 시작하는지 비교	"Apple"로 시작하는 값 검색
EndsWith	특정 값으로 끝나는지 비교	"Apple"로 끝나는 값 검색
LE	값이 작거나 같은지 비교	100 이하의 값 검색
GT	값이 더 큰지 비교	50보다 큰 값 검색
BT	특정 범위 내 값 비교	50과 100 사이 값 검색
NE	값이 다른지 비교	"Apple"이 아닌 값 검색

주로 사용하는 control 별 이벤트 속성

List

- `items`: 데이터를 바인딩할 때 사용됩니다. 보통 `JSONModel`이나 `ODataModel`을 바인딩합니다.
- `headerText`: 리스트의 상단에 표시할 텍스트를 설정합니다.
- `mode`: 리스트 아이템 선택 방식. 예: `"None"`, `"Single"`, `"Multi"`.
- `itemPress`: 리스트 아이템 클릭 시 호출되는 이벤트 핸들러 함수.
- `sorter`: 데이터를 정렬할 때 사용합니다.

ComboBox

- **items** : ComboBox의 항목들을 설정합니다.
- **selectedKey** : ComboBox에서 선택된 항목의 키를 가져옵니다.
- **selectionChange** : 선택이 변경될 때 호출되는 이벤트 핸들러 함수.
- **enabled** : ComboBox의 활성화/비활성화를 설정합니다.
- **editable** : ComboBox가 편집 가능한지 여부를 설정합니다.
-

SearchField

- **search** : 사용자가 검색어를 입력하고 검색을 시작할 때 호출되는 이벤트 핸들러 함수.
- **value** : 검색어를 바인딩합니다.
- **liveChange** : 실시간 검색 필터링을 할 때 사용됩니다.
- **placeholder** : 검색창에 표시될 기본 텍스트를 설정합니다.

StandardListItem

- **title** : 항목의 주 제목을 지정합니다.
- **description** : 항목의 부제목을 지정합니다.
- **type** : 항목의 동작 유형을 정의합니다. 예를 들어, Active는 항목을 클릭할 수 있는 형태로, Navigation은 항목을 클릭하면 네비게이션이 발생하는 형태로 설정할 수 있습니다.
- **press** : 항목을 클릭했을 때 호출되는 이벤트를 처리합니다.