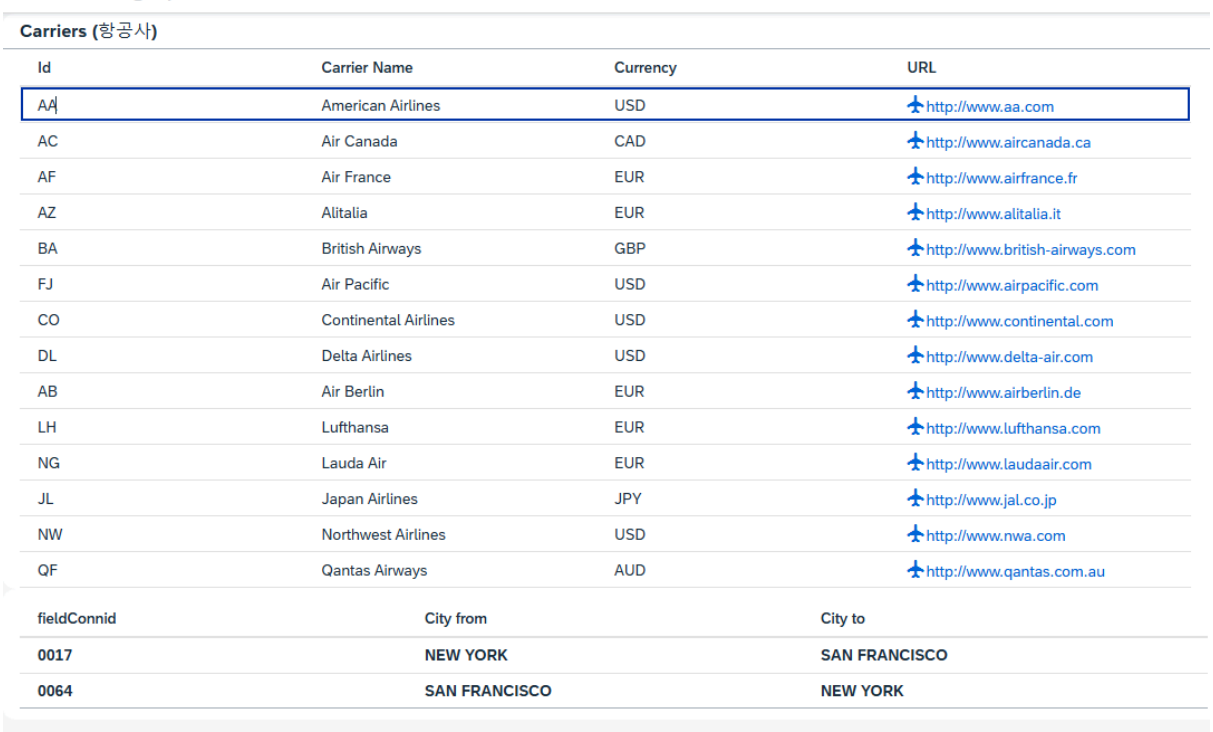
model 및 binding

해당 프로그램을 자세하게 설명한다.

해당 프로그램은 상단의 테이블을 클릭하면

클릭한 행의 항공사가 가지고 있는 연결 정보를

하단의 테이블에 출력한다.



Carriers (항공사)

Id	Carrier Name	Currency	URL
AA	American Airlines	USD	http://www.aa.com
AC	Air Canada	CAD	http://www.aircanada.ca
AF	Air France	EUR	http://www.airfrance.fr
AZ	Alitalia	EUR	http://www.alitalia.it
BA	British Airways	GBP	http://www.british-airways.com
FJ	Air Pacific	USD	http://www.airpacific.com
CO	Continental Airlines	USD	http://www.continental.com
DL	Delta Airlines	USD	http://www.delta-air.com
AB	Air Berlin	EUR	http://www.airberlin.de
LH	Lufthansa	EUR	http://www.lufthansa.com
NG	Lauda Air	EUR	http://www.laudaair.com
JL	Japan Airlines	JPY	http://www.jal.co.jp
NW	Northwest Airlines	USD	http://www.nwa.com
QF	Qantas Airways	AUD	http://www.qantas.com.au

fieldConnid	City from	City to
0017	NEW YORK	SAN FRANCISCO
0064	SAN FRANCISCO	NEW YORK

o 로시작하는 변수들은 OBJECT 객체를 의미

s 로시작하는 변수들은 string 문자열을 의미한다.

model 기초!!!!

model 은 기본적으로 manifest.json 에서 등록이 되면 globar model 로 작동을 한다.

global model도 this.getOwnerComponent().setModel(모댈객체, "모델이름") 로 수동 등록이 가능하다.

view 에 view.setModel(모댈객체, "모델이름") 양식으로 등록이 되면 view의 model로 작동을한다.

table 에 oContext.bindElement(”모델이름”) 같은 양식으로 등록되면 control 의 model 로 작동한다.

모델은 자신과 가장 가까운 지점부터 모델을 적용한다. control 에 모델이 있다면 control에 binding된 model 을 사용, control 에 model 이 없고 view 에 모델이 있으면 view 의 model 을 사용, control 및 view에 model 이 없다면. gloab model 을 사용한다.

setModel() 은 model 전체를 설정하는거고 (모델 객체를 받음 object)

bindElement() 는 model 에서 특정경로를 설정 (경로를 받음 string)

그래서 table 에서 몇번째 행을 눌러서 그 정보를 다른 table 로 넘기려면 bindElement() 를 사용하여 시작할 경로를 줘야한다.

oEvent로 table 이나 control 간 model 을 binding 할때는 bindElement() 를 사용해야 하는 이유를 아래에 예시로 설명을 해주겠다.

bindElement() 를 사용해야 하는경우

데이터의 특정 항목에만 binding을 해야할 경우

사용한 모델의 양식
manifest.json에 data로 등록되어 있는 경우와
빈값 ""이 등록되어 있는 경우 시나리오를 2개 작성한다.


```
{
  "Carrier": [
    {
      "__metadata": {
        "id": "http://bgissap1.bgissap.co.kr:8000/sap/opu/odata/SAP/YC00_CARR
        "uri": "http://bgissap1.bgissap.co.kr:8000/sap/opu/odata/SAP/YC00_CAR
        "type": "YC00_CARR_CONN_SRV.Carrier"
      },
      "Carrid": "AA",
      "Carrname": "American Airlines",
      "Currcode": "USD",
```

```
"Url": "http://www.aa.com",
"toConnection": {
 "results": [
  {
   "__metadata": {
    "id": "http://bgissap1.bgissap.co.kr:8000/sap/opu/odata/SAP/YC00_C
    "uri": "http://bgissap1.bgissap.co.kr:8000/sap/opu/odata/SAP/YC00_(
    "type": "YC00_CARR_CONN_SRV.Connection"
   },
   "Carrid": "AA",
   "Connid": "0017",
   "Countryfr": "US",
   "Cityfrom": "NEW YORK",
   "Airpfrom": "JFK",
   "Countryto": "US",
   "Cityto": "SAN FRANCISCO",
   "Airpto": "SFO",
   "Fltime": "  6:01",
   "Deptime": "PT11H00M00S",
   "Arrtime": "PT14H01M00S",
   "Distance": "2572.000",
   "Distid": "MI",
   "Fltype": "",
   "Period": 0
  },
  {
   "__metadata": {
    "id": "http://bgissap1.bgissap.co.kr:8000/sap/opu/odata/SAP/YC00_C
    "uri": "http://bgissap1.bgissap.co.kr:8000/sap/opu/odata/SAP/YC00_(
    "type": "YC00_CARR_CONN_SRV.Connection"
   },
   "Carrid": "AA",
   "Connid": "0064",
   "Countryfr": "US",
   "Cityfrom": "SAN FRANCISCO",
   "Airpfrom": "SFO",
   "Countryto": "US",
   "Cityto": "NEW YORK",
```

```
        "Airpto": "JFK",
        "Fltime": "  5:21",
        "Deptime": "PT09H00M00S",
        "Arrtime": "PT17H21M00S",
        "Distance": "2572.000",
        "Distid": "MI",
        "Fltype": "",
        "Period": 0
      }
    ]
  }
 },

 ...
```

위 onPress event가 발생한 table

```
<Table
   id="idCarrTable"
   items="{data>/Carrier}" 모델 이름이 등록되어 있을경우
   items="{/Carrier}"      모델 이름이 "" 일경우
>
// data> 이것은 data 이름을 가진 model의 경로에서 데이터를 가져온다는 뜻

!! 절대경로 "/"로 시작하는 경로는 항상 최상위 에서부터 시작
!! 경로가 고정되어 항상 최상위 경로 (예: "/Carrier/2")에서부터 시작
!! 예를 들어, "/Carrier/2"라는 경로가 주어지면 항상 그 경로에서 시작
!! 그 이후 경로를 수정해도 다른 영향을 받지 않는다

!! 그런데 "/" 없이 상대경로를 쓰면, 경로가 현재 바인딩된 경로에서부터 상대적으로 시작
!! 예를 들어, "carrid/2"가 바인딩되어 있으면, 그다음 경로는 "carrid/2/~"처럼 계속 확
!! 그래서 상대경로를 쓰면 현재 경로를 기준으* 새로운 경로가 정해짐

!! 절대경로("/")를 쓰면 경로가 고정되기 때문에 경로 변경에 따른 영향을 최소화할 수 있
!! 상대경로는 현재 경로를 기준으로 경로가 확장
!! 태이블의 클릭한 행의 경로로 부터 다른 테이블을 생성해야한다면, 기존 경로부터 시작
!! 클릭한 태이블의 행의 값을 기준으로 경로를 시작해야함으로 상대경로로 작성한다.
```

```
!! 예시:
!! 절대경로
"/Carrier/2"; // 항상 "/Carrier/2"부터 시작
!! 상대경로
"Carrid/2/~"; // "carrid/2"를 기준으로 경로가 확장

    <columns>
      <Column>
        <ObjectIdentifier title="{i18n>fieldCarrid}" />
      </Column>
      <Column>
        <ObjectIdentifier title="{i18n>fieldCarrname}" />
      </Column>
      <Column>
        <ObjectIdentifier title="{i18n>fieldCurrency}" />
      </Column>
      <Column>
        <ObjectIdentifier title="{i18n>fieldUrl}" />
      </Column>
    </columns>
     !! items 에서 읽은 경로를 한줄씩 출력
     !! /Carrier/1,
     !! /Carrier/2,
     !! /Carrier/3,
     !! 이런식으로 ColumnListItem에 줄단위로 출력이 된다.
    <items>
      <ColumnListItem
        type="Active"
        press=".onPress"
      >
        <Text text="{data>Carrid}" /> // model 명 data
        <Text text="{Carrid}" />      // model 명이 없는 경우
        <Text text="{data>Carrname}" />
        <Text text="{Carrname}" />
        <Text text="{data>Currcode}" />
        <Text text="{Currcode}" />
      </ColumnListItem>
    </items>
```

```
  </Table>
</Panel>
```

table 이나 list 에서 발생한 클릭 이벤트처리
element binding
bindElement() 사용

```
/*
 해당 시나리오는 table 에서 특정 행을 클릭하여 oEvent의 형식으로 발생한 event를 받
 oEvent 는 이벤트가 발생한 control인 ColumnListItem의 전체 정보, ID, 이벤트명 등 다
 내용을 가지고 있다.
*/
onPress(oEvent-이벤트 전체 정보를 담고 있는 객체 ){

    // event 가 발생한 control(ui) 을 가져온다.
    // 여기서 event 가 발생한 control인 ColumnListItem를 가져온다.
    // ColumnListItem중에 클릭이 일어난 행만 가지고 온다.
    var oSelectedItem = oEvent.getSource();

    // 해당 control 에 binding된 Model 정보를 가져온다.
    // oContext 는 모델의 경로 path 와, model 을 가지고 있다.
    // 클릭이 발생한 ColumnListItem 행에 바인딩된 model 정보를 객채의 형식으로 가
    var oContext = oSelectedItem.getBindingContext();
    // 모델명이 등록되어 있을경우 모델이름을 적어준다.
    var oContext = oSelectedItem.getBindingContext("모델이름");

    // Model 경로를 가져온다.
    // 위에서 2번째 행을 클릭했다 가정하자
    // 기본 경로는 /Carrier 이다 그럼으로
    // /Carrier/2/ 를 클릭한게 된다.
    // 즉 현재 클릭한 행의 경로를 가져오는 것이다.
    var sPath = oContext.getPath();

!!시나리오 1. 하단의 table contorl에 직접 model 을 binding

    // table control 에 id 기준으로 전달 할 수 도 있다.
        // 왜?? 하단의 테이블에 model을 전달하려면 하단 table의 객체정보가 필요
    var oContext = this.byId("table control 의 id");
```

```
    // 하단 table의 객채에 경로 "/Carrier/2/"로 부터 시작하는 model을 바인딩 한다.
    // 모델중 /Carrier/2/~ 의 하위 경로에 해당하는 model 만 바인딩된다.
    // 그러면 하단 table 의 경로는 /Carrier/2/~ 로 부터 시작한다.

    // 모델 이름이 있는경우 바인딩
    oContext.bindElement({
        path: sPath,         // 바인딩 경로
        model: "data"      // 사용하려는 모델 이름
        });
    // 모델 이름이 없는경우
    oContext.bindElement(sPath);

!!시나리오 2. view에 직접 model을 binding
        // 현재 화면의 view 객체를 가지고온다.
    var oView = this.getView();
    // view 객체에 spath 경로부터 시작하는 model 을 binding 한다.
    oView.bindElement({
        path: sPath,         // 바인딩 경로
        model: "data"      // 사용하려는 모델 이름
        });

    oView.bindElement(sPath);


  },
```

table 이나 list 에서 발생한 클릭 이벤트처리
element binding
setModel() 사용

```
onPress(oEvent) {
        // 이 과정을 같으니 위에 참고
    var oSelectedItem = oEvent.getSource();
    var oContext = oSelectedItem.getBindingContext();
    var sPath = oContext.getPath();

        // 현재 view에서 table 에 사용중인 data 라는 이름의 model의 객채를 가져온다.
```

```
    var oModel = this.getView().getModel("data");
    // oModel 에서 sPath 경로에 있는 데이터들만 가져온다.
    // {
                a:1,
                b:2
        } 이런 양식으로 데이터를 객체 형식으로 반환
    var oData = oModel.getProperty(sPath);

        // 가져온 oData  데이터의 양식으로 다시 모델은 만든다.
    var oNewModel = new JSONModel(oData );
    // 다시 만든 모델을 table control 에 binding 한다.
    this.byId("table control 의 id").setModel(oNewModel);
}
```

비효율적인것이 느껴지는가?
모델을 해체하고, 경로를 가지고온다음, 다시 모델을 만들고 모델을 등록한다.
비효율적

여기서 왜 view에 직접 binding을 하는데 상단의 table에 출력되는 내용은 안바뀌어요 ????

상단의 table은 절대경로로 지정되어 있기때문에 항상 고정된 경로 값만 출력된다. 만약 동일한 이름을 가지고 전혀 다른 양식의 model이 바인딩이되면 그 모델에 table 에 출력해야 하는 경로가 없으면 nodata 즉 빈칸들이 나오게 된다.

하단 table 의 xml 이다.

```
<Table
      id="idConnTable"
     items="{toConnection/results}" !! 상대경로이다.
     items="{data>toConnection/results}" !! 만약 model 이름이 있는경우
     >
     !! 2 경우 모두 경로 앞에 /Carrier/2/ 가 생략되어 있다.
     !! 왜냐?? spath 로 받은 경로값 /Carrier/2 가 상대경로로 설정이 되어 있어서
     !!/Carrier/2 부터 경로가 확장된다.
     !! 즉 /Carrier/2/toConnection/results 이부분 부터 loop 를 돌며 한줄씩 출력한다.
```

```
    <columns>
      <Column>
        <Text text="{i18n>fieldConnid}" />
      </Column>
      <Column>
        <Text text="{i18n>fieldCityfrom}" />
      </Column>
      <Column>
        <Text text="{i18n>fieldCityto}" />
      </Column>
    </columns>
    <items>
      <ColumnListItem>
        <ObjectIdentifier title="{Connid}" />  // 모델 이름이 없는경우
        <ObjectIdentifier title="{data>Connid}" /> // 모델 이름이 있는경우
      </ColumnListItem>
    </items>
  </Table>
```

setModel()을 사용해야 하는경우

모델 자체를 넘겨줘야 하는

this.getView() 는 현재 화면 view 의 객체를 말한다(view 도 control임 !!)

case1 Json model 을 생성후 model 을 binding

"sap/ui/model/json/JSONModel" 를 define 부분에 등록하고
JSONModel 을 반환하고 사용

```
//
var oData = { test: 1}

// oData 양식의 JsonModel 객체를 생성한다.
let oModel = new JSONModel(oData);
// JsonModel 을 control 객체에 'carr' 이라는 이름으로 등록한다.
```

```
// oContext 는 (this.view, table 등 다양한 control 객체가 올 수 있다.)
this.oContext.setModel(oModel, "carr");

-------------------------------------------------------------------------------
```

case2 view 에 bidinng 된 model 을 table 에 binding (말이 안되지만 예시를 듬)
모델을 control 에서 가져올 수 있다는 것을 보여줌

```
// 현재 view 에서 "data" 라는 이름의 model 객체를 가져온다.
let oModel = this.getView().getModel("data")
// table, list 등 다양한 contorl 에서도 가져올 수 있다.
let oModel = oContext.getModel("data")

// 내가 model 을 바인딩 시키고 싶은 control에 위에서 받아온 model을 data2 라는 이
otargetControl.setModel(oModel ,"data2")


-------------------------------------------------------------------------------
```

응용 press event 말고 rowSelectionChange event를 사용하는 경우 바인딩 하기

시나리오. sap.m.table 이 아닌 sap.ui.table을 사용하는경우
sap.m.table 은 press event 사용 (화면에 따라 반응형으로 화면이 바뀜)
sap.ui.table 은 rowSelectionChange 이벤트 사용 (화면의 크기가 고정되어 있음)
자세한 차이는 다음 블록에서 설명

---

grid table 에서 특정 행을 선택하여 dialog 로 그 행에 해당하는 model 을 보내기

```
async openDialog(oEvent) {
```

```
!! event 가 rowSelectionChange임 event 가 press 가 아니라
!! 왜 getSoruce() 를 안쓰고 getParameter() 냐면 각각의 이벤트가 가지고 있는 나
this.oDialog ??= await this.loadFragment({
  name: "lsyn.c15.ui5.basic.hw1.view.Info",
});
// 내가 선택한 행의 정보를 getParameter() 에서 "rowContext" 로 가져온다.
// rowSelectionChange 의 parameter 참조
// https://sapui5.hana.ondemand.com/#/api/sap.ui.table.Table%23events
// rowSelectionChange 는 rowContext 사용하여 행 정보를 받아 올 수 있다.
var oContext = oEvent.getParameter("rowContext");

// 행의 model 주소를 가져온다.
var sPath = oContext.getPath();

// 현재 view에서 사용한 modeld을 받아온다.
let oModel = this.getView().getModel("book2");

// 받아온 모델에서, 내가 클릭한 경로에 있는 proerty(해당하는 위치에 있는 필드값
// { test: 1. test2: 2} 이런식으로 반환.
let oSelectedData = oModel.getProperty(sPath);

// dialog에 model 을 설정하기 위해 결과 값을 jsonmodel로 만든다.
// oSelectedData 는 oData 가 아니라 단순 구조체임으로 oData 로 변경해준다.
// oData 는 객체 object 이다.
// dialog 에 다시 만든 model을 등록한다.
this.oDialog.setModel(new JSONModel(oSelectedData), "book3");

this.getView().addDependent(this.oDialog);

// console.log(oSelectedData);

this.oDialog.open();

// // 모델이 잘 binding 되었나 확인
// // carr 이라는 이름의 model 의 객채를 가져온다.
// const oDialogModel = this.oDialog.getModel("carr");
// console.log(oDialogModel); // 모델 출력
// console.log(oDialogModel.getData()); // 모델의 데이터 출력
```

```
    /* getProperty 를 활용하는 방법
            <Text id="CarridText" />  처럼 model 이 아니라 id 기준으로 값을 직접
            // oData model 에서 .getProperty(경로) 를 사용하여 경로에 해당하는 값
            // 여기서는  가져올 수 있다. oModel.getProperty("/Carrid") 가 string이
            // 받는 데이터가 구조체면 구조체를 반환함
            var sCarrid = oModel.getProperty("/Carrid");
            var oCarridText = this.oDialog.byId("CarridText");
            oCarridText.setText(sCarrid);
            이런식으로 작성
            */
    },
```

sap.m.table 과 sap.ui.table 의 차이

GPT 로 요약했다....

- *sap.m.Table** 과 *sap.ui.table.Table** 은 SAPUI5에서 두 가지 종류의 테이블 컨트롤입니다. 각 테이블은 서로 다른 목적과 특성을 가지고 있으며, 이를 바탕으로 어떤 상황에서 사용할지 결정할 수 있습니다. 아래는 두 테이블의 주요 차이점과 각 테이블을 사용할 때의 가이드 코드입니다.

1. sap.m.Table vs sap.ui.table.Table

특성	sap.m.Table	sap.ui.table.Table
사용 목적	모바일 및 데스크탑을 위한 경량화된 테이블	데스크탑 중심의 고성능 테이블
성능	상대적으로 성능이 낮고, 많은 양의 데이터에 비효율적일 수 있음	많은 양의 데이터와 복잡한 기능에 적합
지원하는 기능	기본적인 테이블 기능 (스와이프, 터치 등)	고급 기능 (헤더 고정, 열 필터링, 정렬, 페이징 등)
컨트롤	다양한 m 라이브러리 요소 (버튼, 스위치 등)와 함께 사용할 때 더 적합	복잡한 데이터 바인딩과 같은 고급 기능을 필요로 할 때 사용
반응형 디자인	모바일 친화적이고 반응형 디자인 제공	반응형 디자인 미지원 (하지만, 여러 기능으로 커스터마이징 가능)

2. 사용 사례

- **sap.m.Table**:

- 모바일 UI와 반응형 UI에 적합.
- 기본적인 리스트나 간단한 데이터를 표현하는 데 적합.
- 예시: 모바일에서 제품 리스트, 연락처 목록 등.

- **sap.ui.table.Table**:
  - 데스크탑에서 고성능 테이블을 필요로 하는 경우 사용.
  - 대량의 데이터 처리와 고급 기능(정렬, 필터링, 페이징 등)이 필요한 경우.
  - 예시: 대시보드 테이블, 금융 데이터 테이블, 보고서 테이블 등.

3. 가이드 코드

sap.m.Table 가이드 코드

```
<mvc:View
  controllerName="lsyn.c15.ui5.basic.hw1.controller.Main"
  xmlns:mvc="sap.ui.core.mvc"
  xmlns="sap.m"
  xmlns:i="sap.ui.model.odata.v2">

 <Page title="{i18n>title}">
  <Panel headerText="고객정보" height="40%" width="auto">
   <Table items="{/customerData}" press=".onPress">
    <columns>
     <Column>
      <Text text="고객명" />
     </Column>
     <Column>
      <Text text="이메일" />
     </Column>
    </columns>
    <items>
     <ColumnListItem>
      <Text text="{name}" />
      <Text text="{email}" />
     </ColumnListItem>
    </items>
   </Table>
```

```
    </Panel>
  </Page>
</mvc:View>
```

설명:

- `sap.m.Table`은 모바일 장치나 작은 화면에서의 사용을 고려하여 UI를 구성합니다. 기본적인 데이터를 나열하는 데 적합합니다.
- `items`를 사용하여 테이블의 데이터와 바인딩하며, `ColumnListItem`을 사용하여 각 행을 정의합니다.
- `press` 이벤트는 `onPressTable`이라는 메서드와 연결됩니다. 사용자가 행을 클릭하면 이 메서드가 호출됩니다.

`sap.ui.table.Table` 가이드 코드

```
<mvc:ViewcontrollerName="lsyn.c15.ui5.basic.hw1.controller.Flight"
    xmlns:mvc="sap.ui.core.mvc"
    xmlns="sap.m"
    xmlns:t="sap.ui.table">

  <Page title="{i18n>title}">
    <t:Table rows="{book2>/results}"
      selectionBehavior="Row"
      selectionMode="Single"
      rowSelectionChange=".onRowSelectionChange"> // press 아님 주의
                    <t:extension> // 상단의 확장영역 보통 toolabr 를 넣는다.
          <OverflowToolbar> // 상단의 toolbar
            <Title text="학생정보 " />
          </OverflowToolbar>
        </t:extension>
        <t:columns> // columen 각 column 별, label 과 template 로 구성
          <t:Column>
            <t:label>
              <Title text="학생번호" />
            </t:label>
            <t:template>
```

```
                <ObjectIdentifier title="{student>Stdid}" />
            </t:template>
          </t:Column>
          <t:Column>
            <t:label>
              <Title text="학생이름" />
            </t:label>
            <t:template>
              <Input
                id="idInput"
                value="{student>Name}"
                type="Text"
              />
            </t:template>
          </t:Column>
        <t:columns>
    </t:Table>
  </Page>
</mvc:View>
```

Grid table 기준

template은 하위 항목을 0, 1개 만 가질 수 있다.
그럼으로 <Box> 를 만들어 하위 경로를 1개를 만들고 그안에
다른 요소들을 넣어, 여러가지 항목을 넣도록 한다.

```
  <t:Column>
      <!-- control, ui element 대문자로 시작-->
      <t:label>
        <!-- Aggregation: tColum의 하위속성, 소문자로 시작한다.-->

        <Text text="{i18n>columnUpdate}" />
        <!-- control, UI element 대문자로 시작-->
      </t:label>
      <t:template>
        <!-- Aggregation: t:Column의 하위속성으로 소문자로 시작-->
```

```
        <VBox>
          <DatePicker
            value="{
            path:'Aedat',
            type:'sap.ui.model.odata.type.Date',
            formatOptions: {pattern: 'yyyy.MM.dd'}}"
            displayFormat="yyyy-MM-dd"
            editable="false"
          />
          <TimePicker
            value="{
            path:'Aezet',
            type:'sap.ui.model.odata.type.Time',
            formatOptions: {pattern: 'HH.mm.ss'}}"
            editable="false"
          />
          <Input
            value="{Aenam}"
            editable="false"
          />
        </VBox>
      </t:template>
    </t:Column>
```

설명:

- `sap.ui.table.Table`은 복잡한 데이터 및 고급 기능을 처리할 때 유용합니다.
- `selectionBehavior`와 `selectionMode`를 설정하여 사용자가 행을 선택할 수 있도록 하며, `rowSelectionChange`를 통해 선택된 데이터를 가져옵니다.
- 이 테이블은 고급 필터링, 정렬, 페이징 등의 기능을 지원합니다.

4. 주요 차이점

- **성능**: `sap.m.Table`은 간단하고 가벼운 테이블로 모바일 장치와 반응형 화면에서 잘 작동하지만, 많은 데이터가 있을 경우 성능이 떨어질 수 있습니다. 반면, `sap.ui.table.Table`은 대량의 데이터와 복잡한 작업을 처리하는 데 적합합니다.
- **기능**: `sap.ui.table.Table`은 헤더 고정, 열 필터링, 정렬, 페이징과 같은 고급 기능을 제공하지만, `sap.m.Table`은 간단한 데이터 바인딩과 리스트 형식의 기본적인 기능을 제공합니

다.

5. 선택 기준

- **모바일 및 반응형 디자인**: `sap.m.Table`을 사용.
- **복잡한 데이터 처리 및 고급 기능**: `sap.ui.table.Table`을 사용.