记录编号：_____

嵌入式应用开发
电影排行榜实验指导手册

版本：V 1.0

目录

(一) 实验目的

1. 了解条件渲染，循环渲染
2. 掌握@State、@Prop、@Link，@Builder 装饰器

(二) 实验涉及知识点

1. 数组的使用
2. ForEach 用法
3. 数据源使用
4. List 组件
5. 应用程序上下文
6. 组件封装

(三) 实验准备

1. 技能要求：

操作此实验需要具备基本的 TS 语法知识；

2. 实验环境要求：

基于 Windows10 或者 MacOS 操作系统，安装了 DevEco Studio 开发工具。

(四) 实验内容及课时分配

序号	实验内容	实验课时	对应核心知识点	对应实验目标点
1	电影列表布局	1	1、2、3、4	1、2
2	刷新逻辑实现	1	5、6	1、2
	总计	2		

(五) 实验过程

1. 实验功能描述

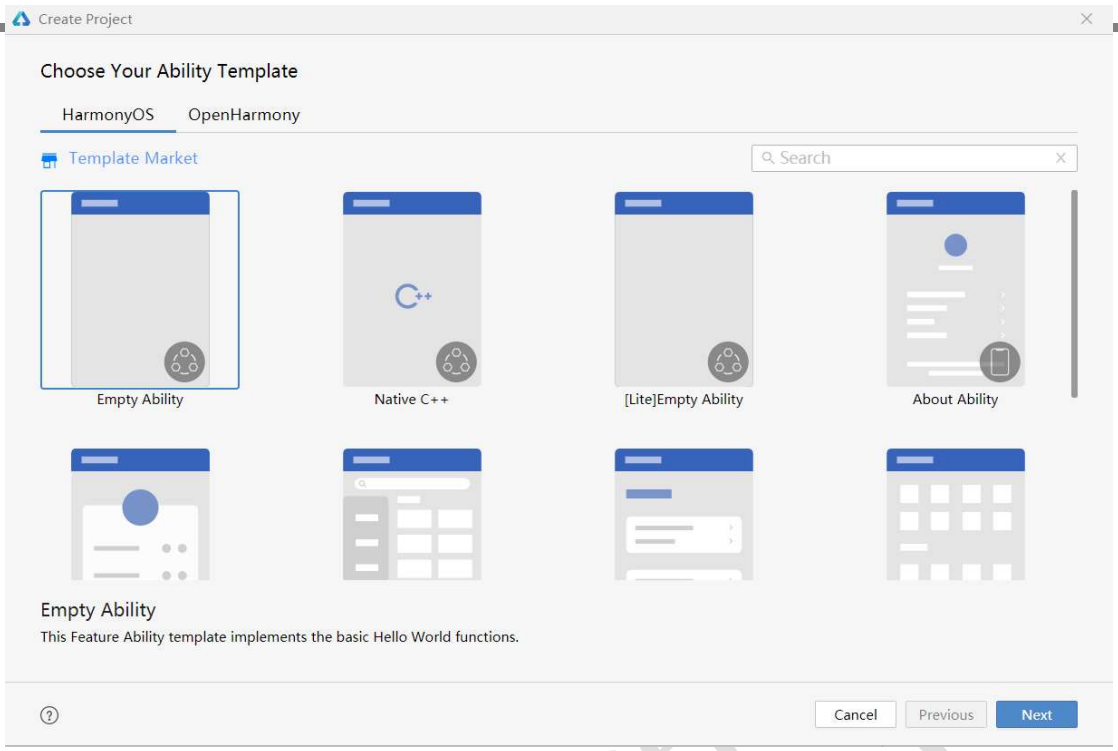
展示电影排行榜

页面效果：

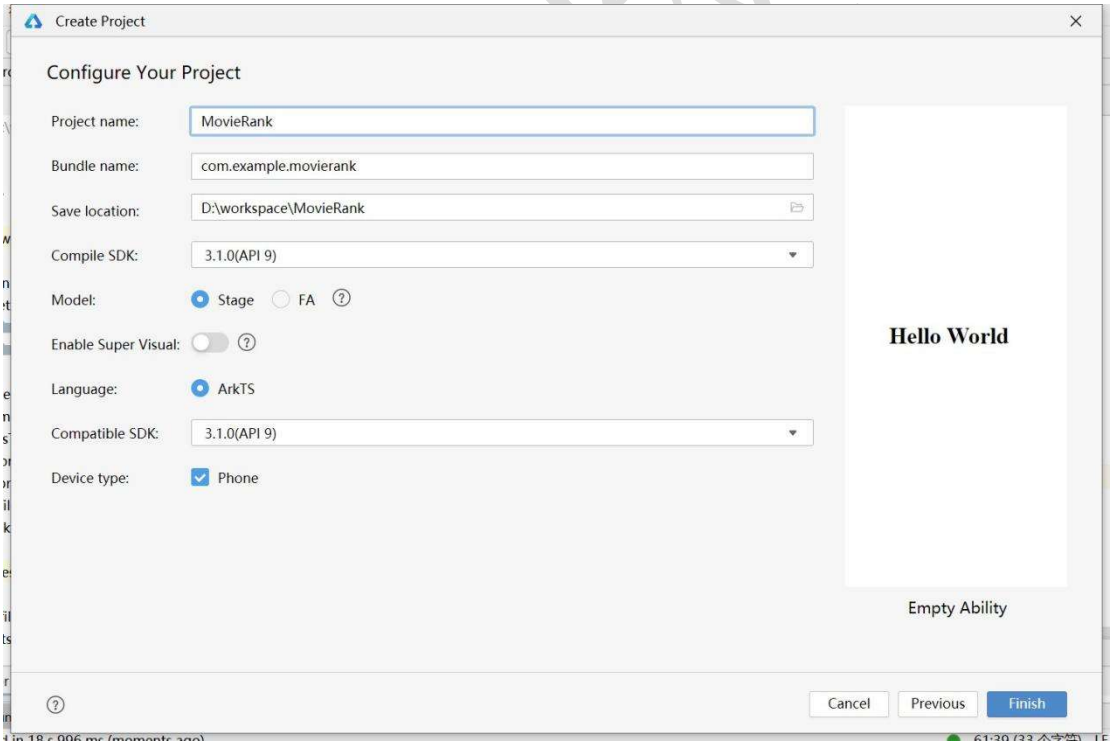
点击返回，关闭窗口，点击刷新，刷新数据，能够显示 top3 影片

2. 新建项目

从 file>new>create project 菜单下新建项目：



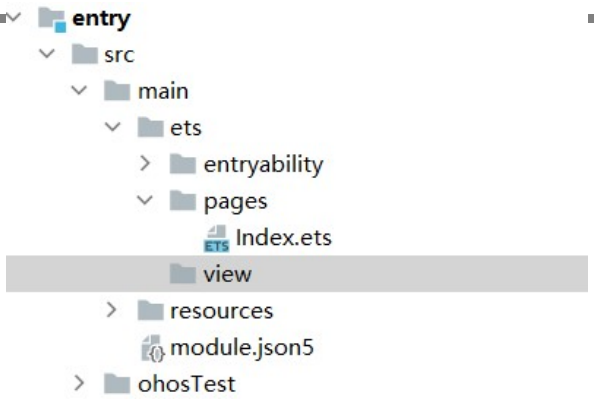
点击【Next】按钮，输入项目配置信息：



点击【Finish】按钮完成项目创建。

3. 封装标题组件

在 ets 目录下新建 view 文件夹用于存放页面组件：



1) 新建 ArkTS 文件

在 view 文件下新建 ArKTS 文件，命名为 TitleComponent.ets

编写基础代码：

```
@Component
export struct TitleComponent {

  build(){

  }

}
```

2) 使用行布局布局组件

```
@Preview
@Component
export struct TitleComponent {
  build() {
    Row() {
      Row() {
        Image($r("app.media.arrowcircleleft"))
          .width(30).height(30)
          .margin({ right: 5 })
        Text('电影排行榜').fontSize(20)

      }.width('50%').height(30).justifyContent(FlexAlign.Start)

      Row(){
        Image($r('app.media.redo'))
          .width(25).height(25)
      }.width('50%')
      .height(30)
      .justifyContent(FlexAlign.End)
    }.width('100%')
```

```
      .padding({left:25,right:25})
      .margin({top:10})
      .height(45)
      .justifyContent(FlexAlign.SpaceAround)
  }
}
```

添加@Preview 装饰器可以在预览器中预览：

3) 在入口页面导入组件并预览效果

编辑 Index.ets 文件

导入组件

```
import { TitleComponent } from '../view/TitleComponent';
```

在 build 中使用组件：

```
import { TitleComponent } from '../view/TitleComponent';
@Entry
@Component
struct Index {


  build() {
    Column(){
      TitleComponent()
    }.width('100%').height('100%')
  }
}
```

预览效果：

4. 封装列表表头

同样在 view 文件夹下新建 ArkTS 文件并命名为：ListHeaderComponent.ets

1) 添加如下基础代码：

```
@Preview
@Component
export struct ListHeaderComponent {

  build(){

  }
}
```

2) 仍然使用 Row 布局布局组件

添加表头，包括排名，影片名，票房。

ListHeaderComponent.ets 中的代码：

```
@Preview
@Component
export struct ListHeaderComponent {

  build(){
    Row(){
      Text('排名').fontSize(20)
        .width('30%')
        .fontWeight(400)
        .fontColor(0xd3d3d3)
      Text('影片名').fontSize(20)
        .width('50%')
        .fontWeight(400)
        .fontColor(0xd3d3d3)
      Text('票房').fontSize(20)
        .width('20%')
        .fontWeight(400)
        .fontColor(0xd3d3d3)
    }.width('90%')
    .margin({left:15,right:15})

  }
}
```

在 Index.ets 中调用组件：

```
import { ListHeaderComponent } from '../view/ListHeaderComponent';
import { TitleComponent } from '../view/TitleComponent';
```

```
@Entry
@Component
struct Index {


  build() {
    Column(){
      TitleComponent()
      ListHeaderComponent()
    }.width('100%').height('100%')
  }
}
```

预览效果：

5. 封装列表项

1) 在 view 文件夹下新建 ArkTS 文件并命名为：ListItemComponent.ets

建立基础代码：

```
@Preview
@Component
export struct ListItemComponent {

  build(){

  }
}
```

2) 布局列表项

还是使用 Row 布局，排布排名，影片名，票房数，先在头部定义变量，然后布局各组件。

```
@Preview
@Component
export struct ListItemComponent {
  index: number = 1
  movieName:string = '满江红'
  tickets:string = '12345'

  build(){
    Row(){
      Column(){
        Text(this.index.toString()).lineHeight(25)
          .textAlign(TextAlign.Center)
          .fontSize(15)
          .width(24)
          .fontWeight(400)
      }.width('30%')
      .alignItems(HorizontalAlign.Start)

      Text(this.movieName)
        .width('50%')
        .fontWeight(500)
        .fontSize(16)
        .fontColor(Color.Blue)

      Text(this.tickets)
        .width('20%')
        .fontWeight(500)
        .fontSize(16)
        .fontColor(Color.Blue)

    }

  }

}
```

在 Index.ets 中调用组件：

```
import { ListHeaderComponent } from '../view/ListHeaderComponent';
import { TitleComponent } from '../view/TitleComponent';
import { ListItemComponent } from '../view/ListItemComponent';
@Entry
```

```
@Component
struct Index {


  build() {
    Column(){
      TitleComponent()
      ListHeaderComponent()
      ListItemComponent()
    }.width('100%').height('100%')
  }
}
```

预览效果：（注意需在 index.ets 中先尝试调用组件并刷新预览器）

3) 排名前三加圈显示

在 ListItemComponent 中首先根据索引判断是否加圈显示：

```
isCircleText(): boolean {
  if(this.index === 1
  || this.index === 2
  || this.index === 3 ){
    return true
  }
  return false
}
```

加圈显示：

```
@Builder CircleText(index:number){
  Row(){
    Text(this.index.toString())
      .fontSize(15)
      .fontWeight(400)
      .fontColor(Color.White)
  }.justifyContent(FlexAlign.Center)
```

```
    .borderRadius(25)
    .width(25)
    .height(25)
    .backgroundColor(Color.Blue)
}
```

条件渲染：

```
Column(){
  if(this.isCircleText()){
    this.CircleText(this.index)
  }else{
    Text(this.index.toString()).lineHeight(25)
      .textAlign(TextAlign.Center)
      .fontSize(15)
      .width(24)
      .fontWeight(400)
  }

}.width('30%')
.alignItems(HorizontalAlign.Start)
```

页面当前代码：

```
@Preview
@Component
export struct ListItemComponent {
  index: number = 1
  movieName:string = '满江红'
  tickets:string = '12345'

  isCircleText(): boolean {
    if(this.index === 1
    || this.index === 2
    || this.index === 3 ){
      return true
    }
    return false
  }
  @Builder CircleText(index:number){
    Row(){
      Text(this.index.toString())
        .fontSize(15)
        .fontWeight(400)
        .fontColor(Color.White)
    }.justifyContent(FlexAlign.Center)
    .borderRadius(25)
```

```
      .width(25)
      .height(25)
      .backgroundColor(Color.Blue)
  }

  build(){
    Row(){
      Column(){
        if(this.isCircleText()){
          this.CircleText(this.index)
        }else{
          Text(this.index.toString()).lineHeight(25)
            .textAlign(TextAlign.Center)
            .fontSize(15)
            .width(24)
            .fontWeight(400)
        }
      }.width('30%')
      .alignItems(HorizontalAlign.Start)

      Text(this.movieName)
        .width('50%')
        .fontWeight(500)
        .fontSize(16)
        .fontColor(Color.Blue)

      Text(this.tickets)
        .width('20%')
        .fontWeight(500)
        .fontSize(16)
        .fontColor(Color.Blue)
    }

  }

}
```

预览效果：

6. 建立排行列表

1) 封装影片数据

对电影排名信息进行封装，在 ets 目录下新建文件夹 common>bean，在 bean 文件夹下创建 ArkTS 文件并命名为 MovieBean：

```
export class MovieBean {
  id:string;
  movieName:string;
  tickets:string;

}
```

2) 定义数据源

在 Index.ets 页面头部引入该 MovieBean 文件，并声明变量，模拟 5 行数据：

```
@State dataScoure1: MovieBean[] =[
  {'id':'1','movieName':'满江红','tickets':'54321'},
  {'id':'2','movieName':'阿凡达','tickets':'44321'},
  {'id':'3','movieName':'你好，李焕英','tickets':'34321'},
  {'id':'4','movieName':'火星救援','tickets':'24321'},
  {'id':'5','movieName':'星球大战','tickets':'14321'}
];
```

3) 建立排行列表（Index.ets）

```
@Builder MovieRankList(){
  Column(){
    List(){
      ForEach(this.dataScoure1,(item,index) =>{
        ListItem(){
```

```
            ListItemComponent({index:index+1,
              movieName:item.movieName,
              tickets:item.tickets})
          }
        },item =>JSON.stringify(item))
      }
      .width('100%')
      .height('70%')
      .divider({strokeWidth:1})
  }.padding(15)
  .borderRadius(15)
  .alignItems(HorizontalAlign.Center)
  .backgroundColor(Color.White)
  .width('90%')
}
```

在 build 中调用并适当调整间距及背景色，Index.ets 中的代码：

```
import { ListHeaderComponent } from '../view/ListHeaderComponent';
import { TitleComponent } from '../view/TitleComponent';
import { ListItemComponent } from '../view/ListItemComponent';
import { MovieBean } from '../common/bean/MovieBean';
@Entry
@Component
struct Index {

  @State dataScoure1: MovieBean[] =[
    {'id':'1','movieName':'满江红','tickets':'54321'},
    {'id':'2','movieName':'阿凡达','tickets':'44321'},
    {'id':'3','movieName':'你好，李焕英','tickets':'34321'},
    {'id':'4','movieName':'火星救援','tickets':'24321'},
    {'id':'5','movieName':'星球大战','tickets':'14321'}
  ];

  @Builder MovieRankList(){
    Column(){
      List(){
        ForEach(this.dataScoure1,(item,index) =>{
          ListItem(){
            ListItemComponent({index:index+1,
              movieName:item.movieName,
              tickets:item.tickets})
          }
        },item =>JSON.stringify(item))
      }
      .width('100%')
```

```
            .height('70%')
            .divider({strokeWidth:1})
        }.padding(15)
        .borderRadius(15)
        .alignItems(HorizontalAlign.Center)
        .backgroundColor(Color.White)
        .width('90%')
    }

    build() {
        Column(){
            TitleComponent()
            ListHeaderComponent()
                .margin({top:10,bottom:10})
            //ListItemComponent()
            this.MovieRankList()
        }.width('100%').height('100%').backgroundColor('#ffa3b8cd')

    }
}
```

预览效果：

7. 加载模拟数据

1) 建立模拟数据

在 ets 下面创建 model 文件夹，并创建 MovieDataModel.ets 文件。

模拟如下数据：

```
//模拟电影排行数据，两个数据源
export {movieData1,movieData2}

const movieData1: object[] = [
  {
    'id':'1',
    'movieName':'满江红',
    'tickets':'99999'
```

```
},
{
  'id':'2',
  'movieName':'阿凡达',
  'tickets':'99910'
},
{
  'id':'3',
  'movieName':'火星救援',
  'tickets':'99905'
},
{
  'id':'4',
  'movieName':'星球大战',
  'tickets':'99900'
},
{
  'id':'5',
  'movieName':'你好，李焕英',
  'tickets':'99899'
},
{
  'id':'6',
  'movieName':'疯狂外星人',
  'tickets':'99799'
},
{
  'id':'7',
  'movieName':'我不是药神',
  'tickets':'99699'
},
{
  'id':'8',
  'movieName':'敢死队',
  'tickets':'99599'
},
{
  'id':'9',
  'movieName':'小猪佩奇',
  'tickets':'99499'
},
{
  'id':'10',
  'movieName':'小羊肖恩',
```

```
    'tickets':'99399'
  }
]

const movieData2: object[] = [
  {
    'id':'11',
    'movieName':'熊出没',
    'tickets':'99299'
  },
  {
    'id':'12',
    'movieName':'功夫熊猫',
    'tickets':'99199'
  },
  {
    'id':'13',
    'movieName':'冰雪奇缘',
    'tickets':'99099'
  },
  {
    'id':'14',
    'movieName':'魔发奇缘',
    'tickets':'98999'
  },
  {
    'id':'15',
    'movieName':'白雪公主',
    'tickets':'97999'
  },
  {
    'id':'16',
    'movieName':'指环王',
    'tickets':'96999'
  },
  {
    'id':'17',
    'movieName':'猫和老鼠',
    'tickets':'95999'
  },
  {
    'id':'18',
    'movieName':'老师，你好',
    'tickets':'94999'
```

```
  },
  {
    'id':'19',
    'movieName':'黑豹',
    'tickets':'93999'
  },
  {
    'id':'20',
    'movieName':'变形金刚',
    'tickets':'92999'
  }

]
```

2) 视图模型层数据模拟

在 ets 下创建 viewmodel，并创建 MovieRankViewModel.ets 文件，从 Model 加载数据到视图模型层：

```
import { movieData2 } from '../model/MovieDataModel';
import { movieData1 } from '../model/MovieDataModel';

export class MovieRankViewModel{

  //加载 movieData1
  loadMovieRankDataSource1():any[]{
    return movieData1;
  }

  //加载 movieData2
  loadMovieRankDataSource2():any[]{
    return movieData2;
  }
}
```

3) 在 Index.ets 中加载模拟数据

导入并声明：

```
import { MovieRankViewModel } from '../viewmodel/MovieRankViewModel';

let movieRankModel: MovieRankViewModel = new MovieRankViewModel()
```

定义 dataScoure1 和 dataScoure2

```
@State dataScoure1: MovieBean[] =[];
@State dataScoure2: MovieBean[] =[];
```

在页面出现时加载数据：

```
aboutToAppear(){
  this.dataScoure1 = movieRankModel.loadMovieRankDataSource1();
  this.dataScoure2 = movieRankModel.loadMovieRankDataSource2();
}
```

当前 Index.ets 中的代码：

```
import { ListHeaderComponent } from '../view/ListHeaderComponent';
import { TitleComponent } from '../view/TitleComponent';
import { ListItemComponent } from '../view/ListItemComponent';
import { MovieBean } from '../common/bean/MovieBean';
import { MovieRankViewModel } from '../viewmodel/MovieRankViewModel';

let movieRankModel: MovieRankViewModel = new MovieRankViewModel()
@Entry
@Component
struct Index {

  // @State dataScoure1: MovieBean[] =[
  //   {'id':'1','movieName':'满江红','tickets':'54321'},
  //   {'id':'2','movieName':'阿凡达','tickets':'44321'},
  //   {'id':'3','movieName':'你好，李焕英','tickets':'34321'},
  //   {'id':'4','movieName':'火星救援','tickets':'24321'},
  //   {'id':'5','movieName':'星球大战','tickets':'14321'}
  // ];

  @State dataScoure1: MovieBean[] =[];
  @State dataScoure2: MovieBean[] =[];

  @Builder MovieRankList(){
    Column(){
      List(){
        ForEach(this.dataScoure1,(item,index) =>{
          ListItem(){
            ListItemComponent({index:index+1,
              movieName:item.movieName,
              tickets:item.tickets})
          }
        },item =>JSON.stringify(item))
      }
      .width('100%')
      .height('70%')
      .divider({strokeWidth:1})
    }.padding(15)
    .borderRadius(15)
    .alignItems(HorizontalAlign.Center)
```

```
      .backgroundColor(Color.White)
      .width('90%')
  }

  aboutToAppear(){
    this.dataScoure1 = movieRankModel.loadMovieRankDataSource1();
    this.dataScoure2 = movieRankModel.loadMovieRankDataSource2();
  }

  build() {
    Column(){
      TitleComponent()
      ListHeaderComponent()
        .margin({top:10,bottom:10})
      //ListItemComponent()
      this.MovieRankList()
    }.width('100%').height('100%').backgroundColor('#ffa3b8cd')

  }
}
```

预览效果，已经可以从 dataSource1 中加载数据：

8. 点击刷新按钮刷新数据

1) 在 Index.ets 中定义变量

```
@State isUpdataDataSource:boolean = true;//是否刷新，用切换数据源模拟
```

2) 在 TitleComponent 组件中定义变量关联父组件变量

注意，一定先注释掉@Preview 装饰器，会和@Link 冲突

```
@Link isUpdataDataSource:boolean;
```

点击标题组件中的刷新图片切换刷新状态：

```
Image($r('app.media.redo'))
  .width(25).height(25)
  .onClick(()=>{
    this.isUpdataDataSource = !this.isUpdataDataSource;
  })
```

3) 在 Index.ets 组件调用时，传递参数

```
TitleComponent({isUpdataDataSource:$isUpdataDataSource})
```

修改 List 组件的 ForEach 代码切换数据源：

```
ForEach(this.isUpdataDataSource?this.dataScoure1:this.dataScoure2,(item,index) =>{
```

当前 TitleComponent.ets 中的代码：

```
// @Preview
@Component
export struct TitleComponent {
  @Link isUpdataDataSource:boolean;
  build() {
    Row() {
      Row() {
        Image($r("app.media.arrowcircleleft"))
          .width(30).height(30)
          .margin({ right: 5 })
        Text('电影排行榜').fontSize(20)

      }.width('50%').height(30).justifyContent(FlexAlign.Start)

      Row(){
        Image($r('app.media.redo'))
          .width(25).height(25)
          .onClick(()=>{
            this.isUpdataDataSource = !this.isUpdataDataSource;
          })
      }.width('50%')
      .height(30)
      .justifyContent(FlexAlign.End)
    }.width('100%')
    .padding({left:25,right:25})
    .margin({top:10})
    .height(45)
    .justifyContent(FlexAlign.SpaceAround)
  }
}
```

当前 Index.ets 中的代码：

```
import { ListHeaderComponent } from '../view/ListHeaderComponent';
import { TitleComponent } from '../view/TitleComponent';
import { ListItemComponent } from '../view/ListItemComponent';
import { MovieBean } from '../common/bean/MovieBean';
import { MovieRankViewModel } from '../viewmodel/MovieRankViewModel';

let movieRankModel: MovieRankViewModel = new MovieRankViewModel()
@Entry
@Component
```

```
struct Index {

  // @State dataScoure1: MovieBean[] =[
  //     {'id':'1','movieName':'满江红','tickets':'54321'},
  //     {'id':'2','movieName':'阿凡达','tickets':'44321'},
  //     {'id':'3','movieName':'你好，李焕英','tickets':'34321'},
  //     {'id':'4','movieName':'火星救援','tickets':'24321'},
  //     {'id':'5','movieName':'星球大战','tickets':'14321'}
  // ];

  @State dataScoure1: MovieBean[] =[];
  @State dataScoure2: MovieBean[] =[];

  @State isUpdataDataSource:boolean = true;//是否刷新，用切换数据源模拟

  @Builder MovieRankList(){
    Column(){
      List(){
        ForEach(this.isUpdataDataSource?this.dataScoure1:this.dataScoure2,(item,index) =>{
          ListItem(){
            ListItemComponent({index:index+1,
              movieName:item.movieName,
              tickets:item.tickets})
          }
        },item =>JSON.stringify(item))
      }
      .width('100%')
      .height('70%')
      .divider({strokeWidth:1})
    }.padding(15)
    .borderRadius(15)
    .alignItems(HorizontalAlign.Center)
    .backgroundColor(Color.White)
    .width('90%')
  }

  aboutToAppear(){
    this.dataScoure1 = movieRankModel.loadMovieRankDataSource1();
    this.dataScoure2 = movieRankModel.loadMovieRankDataSource2();
  }

  build() {
    Column(){
      TitleComponent({isUpdataDataSource:$isUpdataDataSource})
```

```
        ListHeaderComponent()
          .margin({top:10,bottom:10})
        //ListItemComponent()
        this.MovieRankList()
      }.width('100%').height('100%').backgroundColor('#ffa3b8cd')

    }
}
```

4) 测试效果

点击右上角的刷新后：

9. 模拟返回功能

在点击了左上角的返回按钮后，模拟返回功能。

1) 在 TitleComponent.ets 中导入上下文

```
import AppContext from '@ohos.app.ability.common'
```

2) 响应退出操作：

```
Image($r("app.media.arrowcircleleft"))
  .width(30).height(30)
  .margin({ right: 5 })
  .onClick(()=>{
    let handler = getContext(this) as AppContext.UIAbilityContext;
    handler.terminateSelf()
  })
```

3) 效果测试

注意：不要在预览器中模拟，因为启用了上下文，预览器不支持上下文模拟，需要启动远程模拟器进行测试


5:39
100%
电影排行榜
排名
影片名
票房
1 满江红 99999
2 阿凡达 99910
3 火星救援 99905
4 星球大战 99900
5 你好，李焕英 99899
6 疯狂外星人 99799
7 我不是药神 99699
8 敢死队 99599
9 小猪佩奇 99499
10 小羊肖恩 99399