软通教育

智能家居App主页面开发实验指导手册

版本：V 1.0

**北京软通动力教育科技有限公司**

**2024/05/20**

目 录

[**(一) 实验目的 3**](#_Toc21418)

[**(二) 实验涉及知识点 3**](#_Toc29820)

[**(三) 实验准备 3**](#_Toc1845)

[**(四) 实验内容及课时 4**](#_Toc18801)

[**(五) 详细实验过程 4**](#_Toc17848)

[1. “首页(空状态)”页面静态数据呈现 5](#_Toc12638)

[2. “首页”页面静态数据呈现 6](#_Toc4261)

[3. “场景”页面静态数据呈现 9](#_Toc279)

[4. “消息”页面静态数据呈现 11](#_Toc26142)

[5. “我的”页面静态数据呈现 15](#_Toc12388)

[6. “导航栏”自定义组件 19](#_Toc20529)

[(六) 扩展内容20](#_Toc29918)

1. **实验目的**
2. 掌握HarmonyOS移动应用开发工具的使用；
3. 掌握移动应用开发的需求分解与实现步骤拆解；
4. 掌握UI常用组件、线性布局、日志打印、prompt弹窗、页面跳转与传参、条件渲染、循环渲染、自定义组件等等相关开发技术。
5. 锻炼开发文档阅读能力以及知识转化能力；
6. 掌握知识点深入探讨研究、架构设计思维、造轮子思想，通过自定义组件进行锻炼；
7. 养成良好的编程规范，培养清晰的逻辑思维与编程思想；
8. **实验涉及知识点**
9. HarmonyOS移动应用开发工具（DevEco Studio）使用；UI组件使用，包括Row、Column、Text、Image、Button、、ForEach、ListItem、@Extend、Stack、List等；
10. UI布局的使用，主要是线性、层叠布局；
11. 各种事件监听操作与业务逻辑实现；（重难点）
12. 线性布局使用；（重难点）
13. 页面生命周期以及页面之间跳转与传参；
14. 自定义组件以及如何调用；（重难点）
15. 代码编程规范、设计模式；（重难点）
16. **实验准备**
17. 技能要求：

有一定的arkTS编程基础，理解页面生命周期。

1. 实验环境要求：

**本实验基于API9和DevEco Studio 3.1.1 Release**

1. **实验内容及课时**

|  |  |  |
| --- | --- | --- |
| **序号** | **实验内容** | **实验课时** |
| 1 | “首页(空状态)”页面静态数据呈现 | 3 |
| 2 | “首页”页面静态数据呈现 |
| 3 | “场景”页面静态数据呈现 |
| 4 | “消息”页面静态数据呈现 |
| 5 | “我的”页面静态数据呈现 |
| 6 | “导航栏”自定义组件 |

1. **详细实验过程**

完成智慧家居App主页面UI开发，主要效果展示如下图所示：

1. **“首页(空状态)”页面静态数据呈现**



1. 顶部一个Row包含着一个Column和Image，使用Row布局中FlexAlign.SpaceBetween属性使其两端分布。
2. 中间区域由一整个Column包含着Image组件、两个Text组件和一个Button组件，使用Column布局使其垂直平均分布FlexAlign.SpaceEvenly。

具体代码如下：

无设备时的页面布局效果代码如下:

@Component  
**export default struct** Home{  
 build(){  
 Column(){  
 Row(){  
 Column(){  
 Text("我的家庭").fontSize(18).fontWeight(700)  
 Text("共0设备").fontSize(12).fontColor("#999999")  
 }.alignItems(HorizontalAlign.Start)  
  
 Image($r("app.media.add")).width(40)  
 }.width("100%").justifyContent(FlexAlign.SpaceBetween)  
 .height(60).alignItems(VerticalAlign.Center)  
 .padding({left:10,right:10}).backgroundColor("#FFF")  
  
 Column(){  
 Column(){  
 Image($r("app.media.home\_big")).width(110)  
 Text("您还没有智能设备").fontSize(18).fontColor("#666")  
 Text("添加设备后，将为您管理智能家居").fontSize(11).fontColor("#999")  
 Button("立即添加设备").width(180)  
 }.height(300).justifyContent(FlexAlign.SpaceEvenly)  
  
 }.flexGrow(1).width("100%")  
 .justifyContent(FlexAlign.Center)  
 .alignItems(HorizontalAlign.Center)  
  
 }.width("100%").height("100%")  
 .backgroundColor("#F8F8F8")  
 }  
}

1. **“首页”页面静态数据呈现**



1. 中间内容区域首先由条件渲染判断是否有房间数据来决定当前渲染的内容为空或房间列表。
2. 房间列表由循环渲染生成，由原型图可以看出，每一个房间展示的Stack布局一致，故可用循环渲染减少代码量、提高代码可读性。
3. 每一个房间Column组件包含着顶部一个Row（房间名称和设备数量），使用Row布局中FlexAlign.SpaceBetween属性使其两端分布。
4. 底部每一个Row组件包含着一个Image组件和Text组件。
5. 具体代码如下：

房间实体类:

class类:

**export default class** RoomEntity{  
 *//房间名称* roomName:string  
 *//设备数量* sensorNum:string  
 *//温度* temperature:string  
 *//湿度* humidity:string  
 *//亮度* brightness:string  
 *//背景* roomIcon:Resource  
}

数据:写在Home.ets文件里面

**import** RoomEntity **from** "../model/RoomEntity"

@State roomList:Array<RoomEntity> = [  
 {roomName:'室外',sensorNum:'4',temperature:'30',humidity:'49',brightness:'51',roomIcon:$r("app.media.home1")},  
 {roomName:'主卧',sensorNum:'6',temperature:'31',humidity:'47',brightness:'50',roomIcon:$r("app.media.home2")},  
 {roomName:'厨房',sensorNum:'12',temperature:'32',humidity:'46',brightness:'52',roomIcon:$r("app.media.home3")}  
]

当房间设备个数不为0时，需要展示设备信息，设备信息布局可通过List组件实现，并通过forEach进行遍历。当设备个数为0时，则为空状态。因此我们可将Home组件中的设备模板通过if...else进行判断，根据不同的条件展示不同的效果。该模块的实现代码如下:

**if**(**this**.roomList.length !== 0){  
 List(){  
 ForEach(**this**.roomList,(item:RoomEntity)=>{  
 ListItem(){  
 Stack(){  
 Image(item.roomIcon)  
 Column(){  
 Row(){  
 Text(item.roomName).fontSize(16).fontWeight(700).fontColor("#FFF")  
 Text(item.sensorNum+"个设备").fontColor("#FFF")  
 }.width("100%").padding({left:10,right:10})  
 .justifyContent(FlexAlign.SpaceBetween)  
  
 Row(){  
 Row(){  
 Image($r("app.media.u316")).width(30)  
 Text(item.temperature+"℃").fontColor("#FFF").fontWeight(700)  
 }  
  
 Row(){  
 Image($r("app.media.u319")).width(30)  
 Text(item.humidity+"%").fontColor("#FFF").fontWeight(700)  
 }  
  
 Row(){  
 Image($r("app.media.u317")).width(30)  
 Text(item.brightness+"%").fontColor("#FFF").fontWeight(700)  
 }  
 }.width("260").justifyContent(FlexAlign.SpaceBetween)  
 }.width("100%").justifyContent(FlexAlign.SpaceBetween)  
 .height(150).alignItems(HorizontalAlign.Start)  
 }.margin({bottom:5})  
 }  
 })  
 }  
}**else**{

}

1. **“场景”页面静态数据呈现**



1. 顶部标题栏一个Row包含着一个Text组件，使用Row布局使其水平垂直居中排布。
2. 中间内容区域由循环渲染生成，由原型图可以看出，每一个场景展示的Stack布局一致，故可用循环渲染减少代码量、提高代码可读性。
3. 每一个场景Stack包含着一个Image组件和Text组件
4. 文字使用定位使其脱离文档流，从而实现原型图中的效果。

类

**export default class** SceneEntity{  
 text:string  
 imgSrc:Resource  
}

代码

**import** SceneEntity **from** "../model/SceneEntity"  
@Component  
**export default struct** Scene{  
 @State sceneList:Array<SceneEntity> =[  
 {text:'离家时自动打扫',imgSrc:$r("app.media.u1500")},  
 {text:'PM2.5过高时自动开启空净',imgSrc:$r("app.media.u1533")},  
 {text:'回家时自动开启',imgSrc:$r("app.media.u1534")}  
 ]  
  
 build(){  
 Column(){  
 Row(){  
 Text("智慧场景").fontSize(18).fontColor("#6A534D")  
 }.height(50).width("100%").padding({left:10,right:10})  
 .alignItems(VerticalAlign.Center).justifyContent(FlexAlign.Center)  
 .backgroundColor("#FFF")  
  
 List(){  
 ForEach(**this**.sceneList,(item:SceneEntity)=>{  
 ListItem(){  
 Stack({alignContent:Alignment.TopStart}){  
 Image(item.imgSrc).width("100%").height(200)  
 Text(item.text).fontColor("#0000FF").fontSize(16)  
 .fontWeight(800).margin({top:20,left:20})  
 }.margin({top:15})  
 }  
 })  
 }  
 }.width("100%").height("100%").backgroundColor("#f8f8f8")  
 }  
}

1. **“消息”页面静态数据呈现**

1. 顶部tab切换栏，通过循环渲染生成“设备消息”和“系统消息”Text组件，为Text组件绑定点击事件，拿到点击的text的索引值，为点击的一项添加动态样式，从而实现tab切换栏的效果。
2. 中间内容区域通过条件渲染判断当前tab栏选项的索引值，从而达到渲染出“设备消息”列表和“系统消息”列表。
3. 中间列表每一条消息Row，由原型图可以看出，每一条消息展示的Row布局一致，故可用循环渲染减少代码量、提高代码可读性。

具体代码如下：

Tab切换栏：

**const** select\_background\_color = "#FF7733"  
**const** select\_font\_color = "#FFF"  
**const** default\_font\_color = "#000"  
@Component  
**export default struct** Message{  
 @State selectIndex:number = 0  
 @State backgroundColor0:string = ""  
 @State fontColor0:string = ""  
 @State backgroundColor1:string = ""  
 @State fontColor1:string = ""  
  
 changIndex(index:number){  
 **this**.selectIndex = index  
 **this**.backgroundColor0 = ""  
 **this**.fontColor0 = default\_font\_color  
 **this**.backgroundColor1 = ""  
 **this**.fontColor1 = default\_font\_color  
 **if**(index === 0){  
 **this**.backgroundColor0 = select\_background\_color  
 **this**.fontColor0 = select\_font\_color  
 }**else if**(index === 1){  
 **this**.backgroundColor1 = select\_background\_color  
 **this**.fontColor1 = select\_font\_color  
 }  
 }  
 aboutToAppear(){  
 **this**.changIndex(0)  
 }  
  
 build(){  
 Column(){  
 Row(){  
 Row(){  
 Row(){  
 Row(){  
 Text("设备消息").fontColor(**this**.fontColor0)  
 }.flexGrow(1).justifyContent(FlexAlign.Center)  
 .backgroundColor(**this**.backgroundColor0).height("100%")  
 .borderRadius({topLeft:30,bottomLeft:30})  
 .onClick(()=>{  
 **this**.changIndex(0)  
 })  
 Row(){  
 Text("系统消息").fontColor(**this**.fontColor1)  
 }.flexGrow(1).justifyContent(FlexAlign.Center)  
 .borderRadius({topRight:30,bottomRight:30})  
 .backgroundColor(**this**.backgroundColor1).height("100%")  
 .onClick(()=>{  
 **this**.changIndex(1)  
 })  
 }.backgroundColor("#EFEEF4").width(180).height(50)  
 .borderRadius(30)  
 }.flexGrow(1).justifyContent(FlexAlign.Center).margin({left:30})  
  
 Image($r("app.media.setting")).width(30)  
  
 }.width("100%").backgroundColor("#FFF").padding(10)  
  
  
 }.width("100%").height("100%") *// end Column* .alignItems(HorizontalAlign.Center)  
 .backgroundColor("#F8F8F8")  
 }  
}

后面再写：中间内容区域：目前没效果

**import** DeviceMessage **from** "./DeviceMessage"  
**import** SettingMsg **from** "./SettingMsg"

**if**(**this**.selectIndex === 0){  
 DeviceMessage()  
}**else**{  
 SettingMsg()  
}

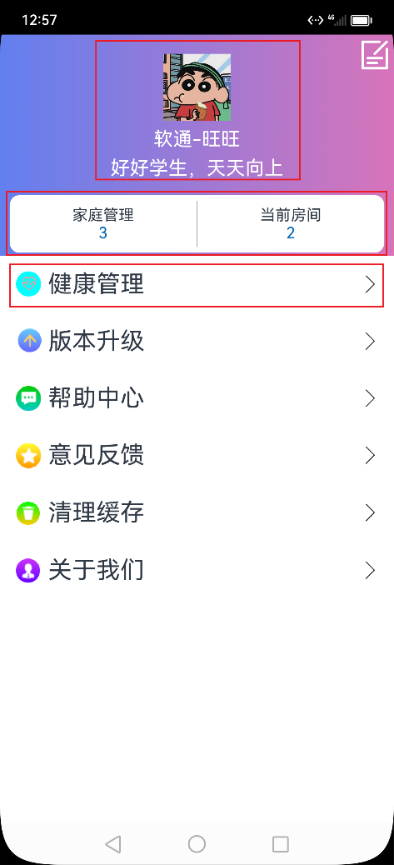
先写这个：设备消息 在DeviceMessage.ets文件里

**class** DeviceMessageEntity{  
 name:string  
 tips:string  
 time:string  
 readFlag:boolean  
}  
@Component  
**export default struct** DeviceMessage{  
 @State messageList:Array<DeviceMessageEntity> = [  
 {name:'监控摄像机',tips:'监测到可疑目标，请及时查看',time:'上午 10:20',readFlag:**false**},  
 {name:'路由器',tips:'网络异常，无法联网，请及时查看',time:'昨天',readFlag:**false**},  
 {name:'监控摄像机',tips:'监测到可疑目标，请及时查看',time:'星期一',readFlag:**true**},  
 {name:'路由器',tips:'网络异常，无法联网，请及时查看',time:'2019/2/11',readFlag:**true**}  
 ]  
 build(){  
 Column(){  
 List(){  
 ForEach(**this**.messageList,(item:DeviceMessageEntity)=>{  
 ListItem(){  
 Column(){  
 Row(){  
 Row(){  
 Text(item.name).fontWeight(700)  
 **if**(!item.readFlag){  
 Text().backgroundColor("#FF7733").width(15).height(15)  
 .borderRadius(15).margin({left:10})  
 }  
 }  
  
 Text(item.time).fontColor("#A7A7A7")  
 }.width("100%").justifyContent(FlexAlign.SpaceBetween)  
 Row(){  
 Text(item.tips)  
 }.width("100%")  
 }.width("95%").padding(10).backgroundColor("#FFF")  
 .margin({ top:10 }).borderRadius(10)  
 }*//end ListItem* }) *// end ForEach* }  
 }.flexGrow(1).width("100%").alignItems(HorizontalAlign.Center)  
 }  
}

系统消息

**class** SettingMsgEntity{  
 title:string  
 text:string  
 time:string  
 imgSrc:Resource  
}  
@Component  
**export default struct** SettingMsg{  
 @State settingMsgList:Array<SettingMsgEntity> = [  
 {title:'今天又有优惠活动喔！',text:'新品家居上市，买一送二，绝不后悔！',time:'04月18日',imgSrc:$r("app.media.messageImg1")},  
 {title:'今天又有优惠活动喔！',text:'新品家居上市，买一送二，绝不后悔！',time:'04月19日',imgSrc:$r("app.media.messageImg2")}  
 ]  
 build(){  
 Column(){  
 List(){  
 ForEach(**this**.settingMsgList,(item:SettingMsgEntity)=>{  
 ListItem(){  
 Column(){  
 Text(item.time).margin({top:10,bottom:10})  
 .fontColor("#A7A7A7")  
 Column(){  
 Text(item.title)  
 Image(item.imgSrc).width("100%")  
 Text(item.text)  
  
 }.backgroundColor("#FFF").width("100%")  
 .height(220).alignItems(HorizontalAlign.Start)  
 .padding(15).justifyContent(FlexAlign.SpaceEvenly)  
  
  
 }.width("100%").margin({bottom:20})  
 }*// end ListItem* })*// end ForEach* }  
 }.width("100%").height("100%") *// end Column* .alignItems(HorizontalAlign.Center)  
 .backgroundColor("#F8F8F8")  
 }  
}

1. **“我的”页面静态数据呈现**



1. 右上角按钮通过Stack层叠组件右上角对齐，实现背景和编辑按钮效果。
2. 头像、昵称及家庭管理区域，使用Column布局使其垂直居中排布。
3. 家庭管理区域中，使用divider组件实现分隔效果。
4. 下方内容区域每一个Row使用线性布局中FlexAlign.SpaceBetween使其两端分布。

具体代码如下：

@Extend(Image) **function** headIcon(){  
 .width(30).margin({right:5})  
}  
@Extend(Text) **function** headText(){  
 .fontSize(25)  
}  
@Extend(Image) **function** headRight(){  
 .width(20)  
}  
@Extend(Row) **function** rowBlock(){  
 .width("100%")  
 .padding({left:15,right:15,top:15,bottom:15})  
 .justifyContent(FlexAlign.SpaceBetween)  
}  
@Component  
**export default struct** MyInfo{  
 build(){  
 Column(){  
 Stack({alignContent:Alignment.Bottom}){  
 Stack({alignContent:Alignment.Top}){  
 Stack({alignContent:Alignment.TopEnd}){  
 Image($r("app.media.color")).width("100%").height(230)  
 Image($r("app.media.edit")).width(40)  
 }.height("100%").width("100%") *//end Stack* Column(){  
 Image($r("app.media.avatar")).width(70).margin({bottom:7})  
 Text("软通-旺旺").fontColor("#FFF").fontSize(20).margin({bottom:7})  
 Text("好好学生，天天向上").fontColor("#FFF").fontSize(20).margin({bottom:7})  
 }.margin({top:20})  
 }.width("100%")*//end Stack* Row(){  
 Column(){  
 Text("家庭管理")  
 Text("3").fontColor("#0064B3")  
 }.flexGrow(1).justifyContent(FlexAlign.Center)  
 .alignItems(HorizontalAlign.Center)  
  
 Row().backgroundColor("#AAA").flexBasis(1).height("80%")  
  
 Column(){  
 Text("当前房间")  
 Text("2").fontColor("#0064B3")  
 }.flexGrow(1).justifyContent(FlexAlign.Center)  
 .alignItems(HorizontalAlign.Center)  
 }.width("95%").height(60).backgroundColor("#FFF")  
 .borderRadius(10).margin({bottom:3})  
 }.height(230) *// end Stack* Row(){  
 Row(){  
 Image($r("app.media.myinfoListicon")).headIcon()  
 Text("健康管理").headText()  
 }  
 Image($r("app.media.right")).headRight()  
 }.rowBlock()  
  
 Row(){  
 Row(){  
 Image($r("app.media.myinfoListicon2")).headIcon()  
 Text("版本升级").headText()  
 }  
 Image($r("app.media.right")).headRight()  
 }.rowBlock()  
  
 Row(){  
 Row(){  
 Image($r("app.media.myinfoListicon3")).headIcon()  
 Text("帮助中心").headText()  
 }  
 Image($r("app.media.right")).headRight()  
 }.rowBlock()  
  
 Row(){  
 Row(){  
 Image($r("app.media.myinfoListicon4")).headIcon()  
 Text("意见反馈").headText()  
 }  
 Image($r("app.media.right")).headRight()  
 }.rowBlock()  
  
 Row(){  
 Row(){  
 Image($r("app.media.myinfoListicon5")).headIcon()  
 Text("清理缓存").headText()  
 }  
 Image($r("app.media.right")).headRight()  
 }.rowBlock()  
  
 Row(){  
 Row(){  
 Image($r("app.media.myinfoListicon6")).headIcon()  
 Text("关于我们").headText()  
 }  
 Image($r("app.media.right")).headRight()  
 }.rowBlock()  
 }.width("100%").height("100%")*// end page* .backgroundColor("#F8F8F8")  
 }  
}

1. **“导航栏”自定义组件**



1. 当页面信息较多时，为了让用户能够聚焦于当前显示的内容，需要对页面内容进行分类，提高页面空间利用率。Tabs组件可以在一个页面内快速实现视图内容的切换，提升查找效率，精简用户单次获取到的信息量。Tabs组件的页面组成包含两个部分，分别是TabContent（内容页）和TabBar（导航页签栏）
2. 点击TabBar时，通过controller控制类切换tab页签
3. Tabs的onChange事件，更新选中的索引。通过点击TabBar（导航页签栏）或左右滑动触发切换页签

具体代码如下：

**import** Home **from** "../components/Home"  
**import** Scene **from** "../components/Scene"  
**import** Message **from** "../components/Message"  
**import** MyInfo **from** "../components/MyInfo"  
@Entry  
@Component  
**struct** Index {  
 @State selectIndex:number = 0  
 controller:TabsController = **new** TabsController()  
  
 @Builder tabBarInfo(index:number,text:string,imageRsc:Resource,activeImage:Resource){  
 Column(){  
 **if**(index === **this**.selectIndex){  
 Image(activeImage).width(30)  
 Text(text).fontSize(18).fontColor("#0099ff")  
 }  
 **else**{  
 Image(imageRsc).width(30)  
 Text(text).fontSize(18).fontColor("#8E8E8E")  
 }  
 }.padding(5).onClick(()=>{  
 **this**.controller.changeIndex(index)  
 })  
 }  
 build() {  
 Tabs({controller:**this**.controller,barPosition:BarPosition.End}){  
 TabContent(){  
 Home()  
 }  
 .tabBar(**this**.tabBarInfo(0,"首页",$r("app.media.tab\_home"),$r("app.media.tab\_home\_active")))  
  
 TabContent(){  
 Scene()  
 }  
 .tabBar(**this**.tabBarInfo(1,"场景",$r("app.media.tab\_scene"),$r("app.media.tab\_scene\_active")))  
  
 TabContent(){  
 Message()  
 }  
 .tabBar(**this**.tabBarInfo(2,"消息",$r("app.media.tab\_message"),$r("app.media.tab\_message\_active")))  
  
 TabContent(){  
 MyInfo()  
 }  
 .tabBar(**this**.tabBarInfo(3,"我的",$r("app.media.tab\_myinfo"),$r("app.media.tab\_myinfo\_active")))  
 }*// end Tabs* .onChange((index)=>{  
 **this**.selectIndex = index  
 })  
 }  
}

1. **扩展内容**
2. 实现样式的方法有很多，实验手册所写样式仅供参考。