《嵌入式应用开发》

青蛙影院项目—首页开发实验指导手册

版本：V 1.0

目录

（一）实验目的

1. 掌握青蛙影院基于 ArkTS 语言的开发。
2. 掌握图片轮播功能的开发。
3. 掌握列表、网格组件的使用。
4. 掌握如何在 ArkTS 语言中使用 MVVM 开发模式进行开发。

（二）实验涉及知识点

1. 基础组件和布局。
2. 装饰器的使用。
3. 图片轮播 Swiper 组件的使用。
4. List 组件的使用。
5. forEach 循环渲染语句。
6. MVVM 开发模式。

（三）实验准备

参考开发环境：

操作系统：Window 10

开发工具：DevEco Studio 3.1.1

HarmonyOS SDK 版本：API version 9 及以上版本

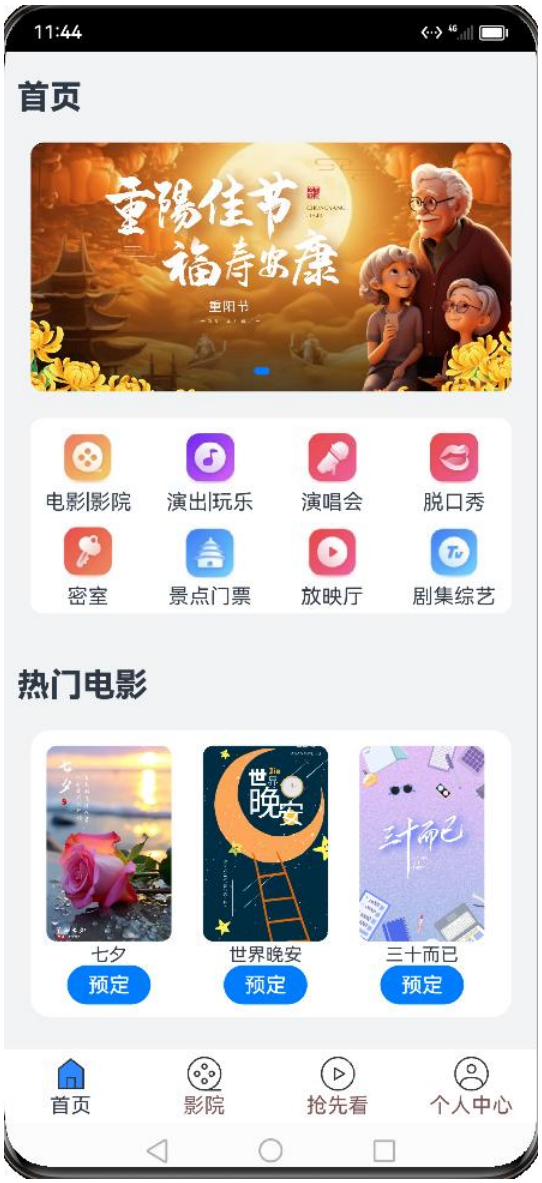
开发语言：ArkTS

内存：8G 及以上

（四）详细实验过程

1 页面布局

首页的最终实现效果图如下图所示。



我们对页面的结构进行分析，从上到下大体分为 3 个部分，轮播图部分、功能菜单部分、以及热门电影部分。

2 页面整体结构与公共标题抽离

在页面布局上，三个部分，每部分都有一个标题，且其样式基本上大同小异，我们可以将该部分抽取成一个单独的组件，实现组件的复用。

打开 Home.ets 文件，编写页面结构及公共标题部分，代码如下所示：

```
@Component
export struct Home {
  // @State message: string = 'page1'

  build() {
    Scroll() {
      Column() {
        //标题 Text
        MyTitle({ title: "首页" })
        //轮播图 Swiper
        MySwiper()
        //功能菜单 Grid
        MyMenu()
        //标题2 Text
        MyTitle({title:"热门电影"})
        //热门电影  List
        MyHotMovie()

      }.width("100%")
      .height("100%").justifyContent(FlexAlign.Start) //有固定的高度才可以设置主轴的对齐方式
    }
  }
}

// 公共标题部分
@Component
```

```
struct MyTitle {
  @State title: string = ""

  build() {
    Text(this.title)
      .fontSize(25)
      .margin({
        top: 20,
        left: 20,
        bottom: 20
      }).width("100%")
      .textAlign(TextAlign.Start)
      .fontWeight(FontWeight.Bolder)
  }
}

//轮播图 Swiper
@Component
struct MySwiper {
  build() {

  }
}

//功能菜单 Grid
@Component
struct MyMenu {
  build() {

  }
}

//热门电影  List
@Component
struct MyHotMovie{
```

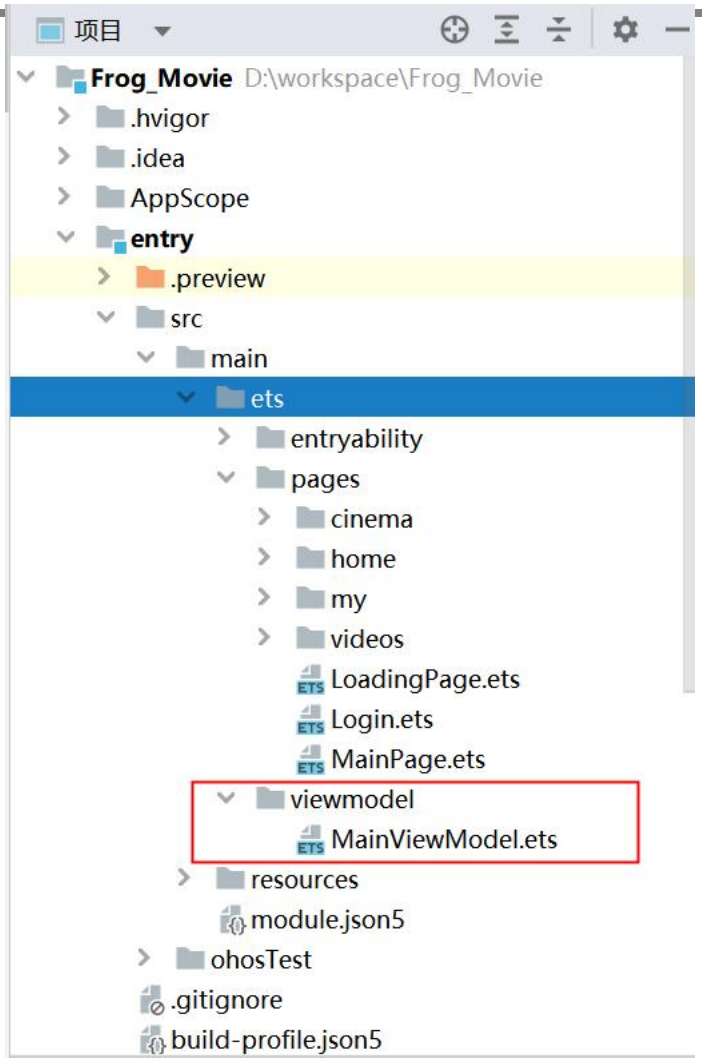
```
  build(){

  }
}
```

3 搭建 ViewModel 框架

为了方便管理我们的页面数据，我们可以采用 MVVM 的开发模式，来实现 View 和 Model 的分离和解耦。

在 viewmodel 包下，我们新建一个 MainViewModel.ets 文件，（如果在实验手册 01-登录页开发的准备工作中，你没有创建该 viewmodel 目录，则这里只需在 ets 目录下再新建目录并命名为 viewmodel 即可。如果已创建了 viewmodel 目录，则可忽略括号中的内容）。创建后的目录结构如下所示：



MainViewModel.ets 中大致的框架如下所示。

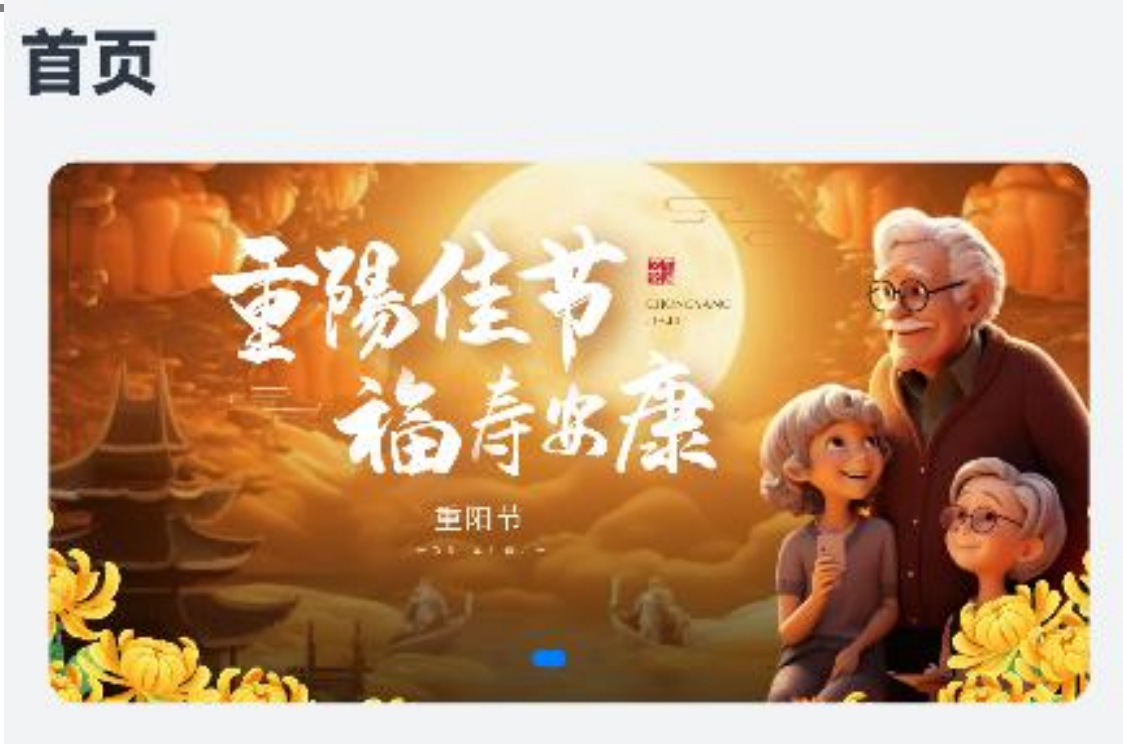
```
export class MainViewModel {


}


export default new MainViewModel()
```

4 轮播图实现

实现的效果图如下所示。



我们在 MainViewModel.ets 中新增提供轮播图的方法，代码如下：

```
export class MainViewModel {

  //电影海报轮播图
  getSwiperImages(): Array<Resource> {
    let images: Array<Resource> = [
    $r("app.media.movie_banner1"),
    $r("app.media.movie_banner2"),
    $r("app.media.movie_banner3"),
    $r("app.media.movie_banner4")
    ]
    return images
  }

}

export default new MainViewModel()
```

在使用的时候，我们一定要在 Home.ets 中导入我们的 ViewModel

```
import viewmodel from '../../viewmodel/MainViewModel'
```

接下来，我们自定义一个组件来实现轮播图，实现轮播图我们采用 Swiper 组件来实现。首先，拿到我们在 ViewModel 中定义的获取数据的方法，得到一个数组，然后通过 ForEach 进行循环渲染，从而得到我们想要的结果。该部分代码写在 Home.ets 的自定义子组件 `MySwiper` 的 `build` 中，代码如下：

```
//轮播图 Swiper
@Component
struct MySwiper {
  build() {
    Swiper() {
      ForEach(viewmodel.getSwiperImages(), (item: Resource, index?: number) =>
{
        Image(item)
          .width("90%")
          .height("25%")
          .borderRadius(10)
      }, item => item.toString())

    }.autoPlay(true)
  }
}
```

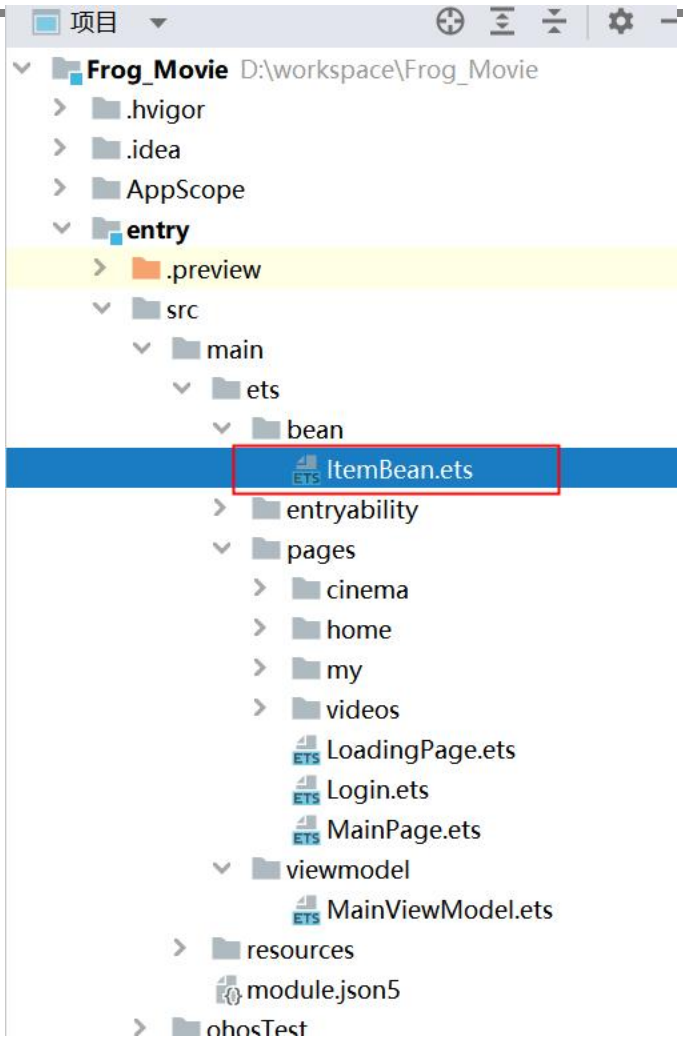
因为我们在搭建页面整体结构的时候已经在 `Entry` 组件中通过 `MySwiper()` 调用了该自定义组件，所以这里只需运行上面的代码即可看到轮播图效果。

接下来继续编写功能菜单部分。

5 功能菜单实现

功能菜单的效果图如下所示。

功能菜单的每一项都有一个 icon 和一段文字描述，对于这种结构非常相似的数据，我们可以定义一个 bean 类，比如我们在 bean 包下新建一个 ets 文件取名为 ItemBean。（如果在实验手册 01-登录页开发的准备工作中，你没有创建该 bean 目录，则这里只需在 ets 目录下再新建目录并命名为 bean 即可。如果已创建了 bean 目录，则可忽略括号中的内容）。创建后的目录结构如下所示：



ItemBean.ets 中的代码如下。

```
export default  class ItemBean{
  name:string
  image:Resource

  constructor(name:string,image:Resource) {
    this.name = name
    this.image = image
  }
}
```

接下来，我们在 ViewModel 中新增提供数据的方法。首先需要在 MainViewModel.ets 导入 ItemBean.ets，然后再添加功能列表数据，代码如下所示。

```
import ItemBean from '../bean/ItemBean'

export class MainViewModel {

  //电影海报轮播图
  ......
  // 常用功能列表数据
  getItemBeanData(): Array<ItemBean> {
    let items: ItemBean[] = [
      new ItemBean("电影|影院", $r('app.media.icon_dianying')),
      new ItemBean("演出|玩乐", $r('app.media.icon_yanchu')),
      new ItemBean("演唱会", $r("app.media.icon_yanchanghui")),
      new ItemBean("脱口秀", $r('app.media.icon_tuokouxiu')),
      new ItemBean("密室", $r('app.media.icon_mishi')),
      new ItemBean("景点门票", $r('app.media.icon_jingdian')),
      new ItemBean("放映厅", $r('app.media.icon_bofang')),
      new ItemBean("剧集综艺", $r('app.media.icon_juji')),
    ]
    return items
  }

}

export default new MainViewModel()
```

然后，我们在首页自定义一个组件来实现我们的功能菜单。在使用的时候，我们也要先在 Home.ets 中导入我们的 ItemBean，导入方法如下：

```
import ItemBean from '../../bean/ItemBean'
```

功能菜单部分代码写在 Home.ets 的自定义子组件 MyMenu()的 build 中，代码如下：

```
//功能菜单 Grid
@Preview
@Component
struct MyMenu {
  build() {
    Grid() {

      //大致结构如下，可以复制 8 份，查看一下效果
      // GridItem(){
      //   Column(){
      //     Image($r('app.media.dianying'))
      //       .width(40)
      //       .height(40)
      //     Text("电影|影院").margin({top:5})
      //   }
      // }
      //使用动态数据
      ForEach(viewmodel.getItemBeanData(), (item: ItemBean, index?: number) =>
{
        GridItem() {
          Column() {
            Image(item.image)
              .width(40)
              .height(40)
            Text(item.name).margin({ top: 5 })
          }
        }
      }, item => item.toString())
    }
    .columnsTemplate("1fr 1fr 1fr 1fr")
    .rowsTemplate("1fr 1fr")
    .rowsGap(15)
    .columnsGap(5)
    .height(150)
    .backgroundColor(Color.White)
```

```
      .borderRadius(10)
      .margin(20)
      .padding({
        top:10,
        bottom:10
      })

  }
}
```

6 横向滚动列表的实现

横向滚动列表的效果图如下所示。

首先，我们需要在 bean 包下定义一个 MovieBean.ets 文件，核心代码如下。

```
export default  class MovieBean{
  name:string
  image:Resource

  constructor(name:string,image:Resource) {
    this.name = name
    this.image = image
  }
}
```

然后，在 ViewModel 中新增获取数据的方法（同样需要在 MainViewModel.ets 中导入 MovieBean.ets），MainViewModel.ets 中横向滚动模块的核心代码如下。

```
import ItemBean from '../bean/ItemBean'
import MovieBean from '../bean/MovieBean'

export class MainViewModel {

  //电影海报轮播图
  ......
  // 常用功能列表数据
  ......
  // 获取热门电影数据
  getHotMovieBeanData(): Array<MovieBean> {
    let movies: MovieBean[] = [
      new MovieBean('七夕', $r('app.media.movie1')),
      new MovieBean('世界晚安', $r('app.media.movie2')),
      new MovieBean('三十而已', $r('app.media.movie3')),
      new MovieBean('你该学习了', $r('app.media.movie4')),
      new MovieBean('节日快乐', $r('app.media.movie5')),
    ]
    return movies
```

```
  }

}

export default new MainViewModel()
```

接下来，我们在首页自定义一个组件实现我们热门电影功能，在使用的时候，我们也要先在 Home.ets 中导入我们的 MovieBean，导入方法如下：

```
import MovieBean from '../../bean/MovieBean'
```

横向滚动列表部分代码写在 Home.ets 的自定义子组件 MyHotMovie()的 build 中，核心代码如下。

```
@Component
struct MyHotMovie{
  build(){
    List() {
      ForEach(viewmodel.getHotMovieBeanData(), (item: MovieBean, index?: numbe
r) => {
        ListItem() {
          Column() {
            Image(item.image)
              .width(96)
              .height(150)
              .borderRadius(8)
              .margin(2)
            Text(item.name)
              .fontSize(14)
            Button("预定")
              .height(30)
          }
        }.margin(10)
      }, item => item.toString())
    }
    .width("90%")
```

```
    .height(220)
    .listDirection(Axis.Horizontal)
    .backgroundColor(Color.White)
    .margin({ left:20, right:20})
    .borderRadius(15)
  }
}
```

7 首页完整代码

首页完整代码如下所示。

```
import viewmodel from '../../viewmodel/MainViewModel'
import ItemBean from '../../bean/ItemBean'
import MovieBean from '../../bean/MovieBean'

@Preview
@Component
export struct Home {
  // @State message: string = 'page1'

  build() {
    Scroll() {
      Column() {
        //标题 Text
        MyTitle({ title: "首页" })
        //轮播图 Swiper
        MySwiper()
        //功能菜单 Grid
        MyMenu()
        //标题2 Text
        MyTitle({title:"热门电影"})
        //热门电影  List
        MyHotMovie()
```

```
      }.width("100%")
      .height("100%").justifyContent(FlexAlign.Start) //有固定的高度才可以设置主
轴的对齐方式
    }
  }
}

// 公共标题部分
@Component
struct MyTitle {
  @State title: string = ""

  build() {
    Text(this.title)
      .fontSize(25)
      .margin({
        top: 20,
        left: 20,
        bottom: 20
      }).width("100%")
      .textAlign(TextAlign.Start)
      .fontWeight(FontWeight.Bolder)
  }
}

//轮播图 Swiper
@Component
struct MySwiper {
  build() {
    Swiper() {
      ForEach(viewmodel.getSwiperImages(), (item: Resource, index?: number) =>
{
        Image(item)
          .width("90%")
          .height("25%")
```

```
            .borderRadius(10)
        }, item => item.toString())

    }.autoPlay(true)
  }
}

//功能菜单 Grid
@Component
struct MyMenu {
  build() {
    Grid() {

      //大致结构如下，可以复制 8 份，查看一下效果
      // GridItem(){
      //   Column(){
      //     Image($r('app.media.dianying'))
      //       .width(40)
      //       .height(40)
      //     Text("电影|影院").margin({top:5})
      //   }
      // }
      //使用动态数据
      ForEach(viewmodel.getItemBeanData(), (item: ItemBean, index?: number) =>
{
        GridItem() {
          Column() {
            Image(item.image)
              .width(40)
              .height(40)
            Text(item.name).margin({ top: 5 })
          }
        }
      }, item => item.toString())
    }
```

```
    .columnsTemplate("1fr 1fr 1fr 1fr")
    .rowsTemplate("1fr 1fr")
    .rowsGap(15)
    .columnsGap(5)
    .height(150)
    .backgroundColor(Color.White)
    .borderRadius(10)
    .margin(20)
    .padding({
      top:10,
      bottom:10
    })

  }
}

//热门电影  List
@Component
struct MyHotMovie{
  build(){
    List() {
      ForEach(viewmodel.getHotMovieBeanData(),  (item:  MovieBean,  index?:
number) => {
        ListItem() {
          Column() {
            Image(item.image)
              .width(96)
              .height(150)
              .borderRadius(8)
              .margin(2)
            Text(item.name)
              .fontSize(14)
            Button("预定")
              .height(30)
          }
```

```
        }.margin(10)
      }, item => item.toString())
    }
    .width("90%")
    .height(220)
    .listDirection(Axis.Horizontal)
    .backgroundColor(Color.White)
    .margin({ left:20, right:20})
    .borderRadius(15)
  }
}
```