《嵌入式应用开发》》

青蛙影院项目—登录页开发实验指导手册

版本：V 1.0

目录

（一）实验目的

1．掌握 HarmonyOS 移动应用开发工具的使用；

2．掌握移动应用开发的需求分解与实现步骤拆解；

3．掌握 Ets UI 组件与布局、监听事件、数据绑定、页面路由、弹窗提示、逻辑算法等等相关开发技术。

4．锻炼开发文档阅读能力以及知识转化能力；

5．养成良好的编程规范，培养清晰的逻辑思维与编程思想；

（二）实验涉及知识点

1. HarmonyOS 移动应用开发工具（DevEco Studio）使用；

2. UI 页面设计原型图分解；

3. UI 组件使用，包括 Text、Button、Image、TextInput 等；

4. 配置文件 config.json 使用；

5. UI 布局的使用，重点涉及 Flex 布局、Row 布局的；

6. 数据绑定以及各种事件监听操作与业务逻辑实现；

7. 日志打印以及查看；

8. 页面路由、弹窗提示等系统 api 使用；

9. 代码编程规范、设计模式；

（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

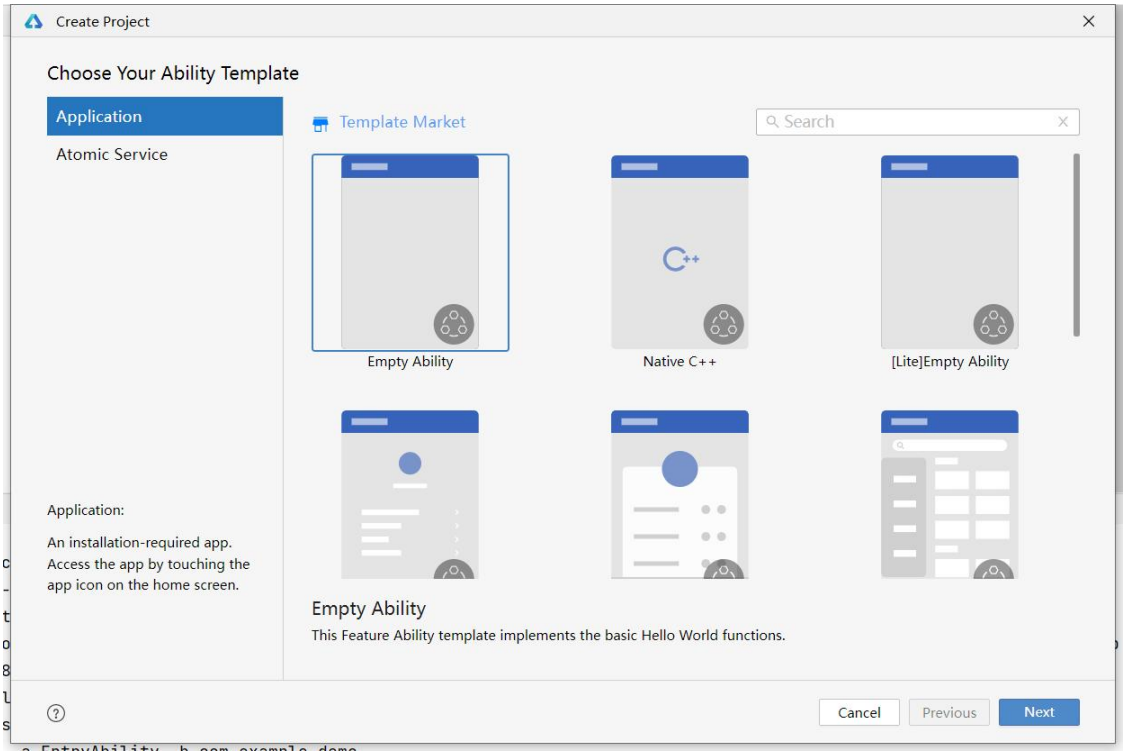
内存：8G 及以上

（四）详细实验过程

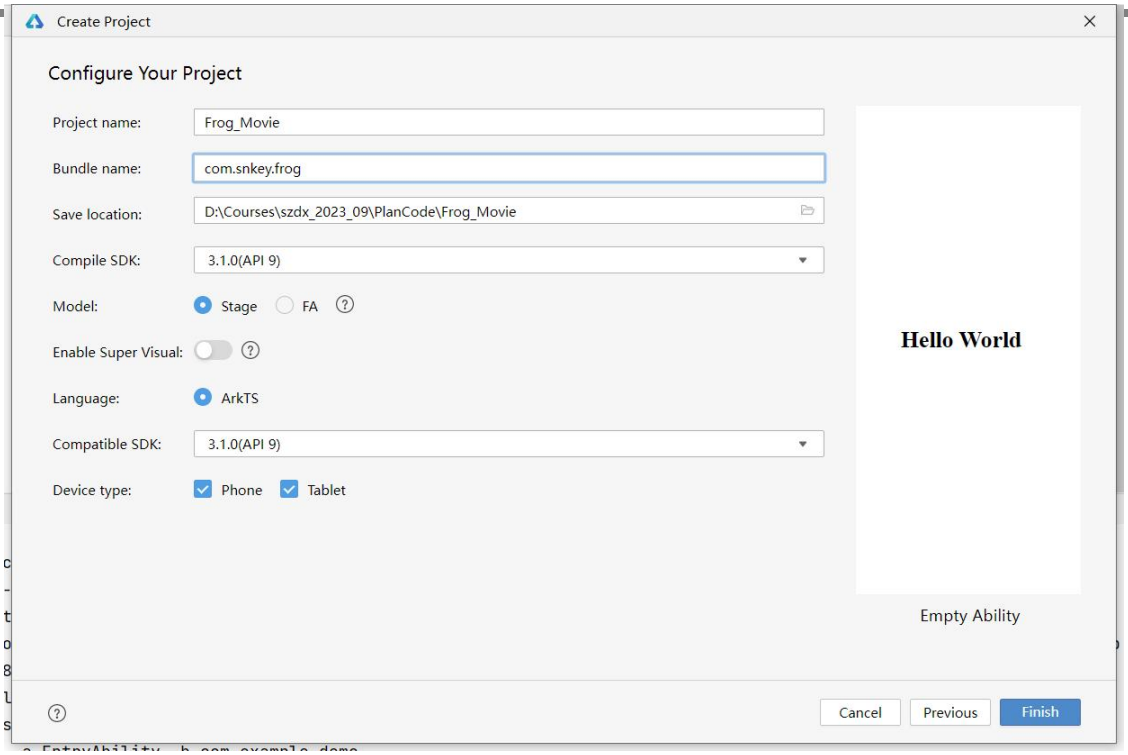
1 准备工作

1.1 新建工程

如下图所示，新建一个工程。



点击“Next”，如下图所示，填写项目基本信息。

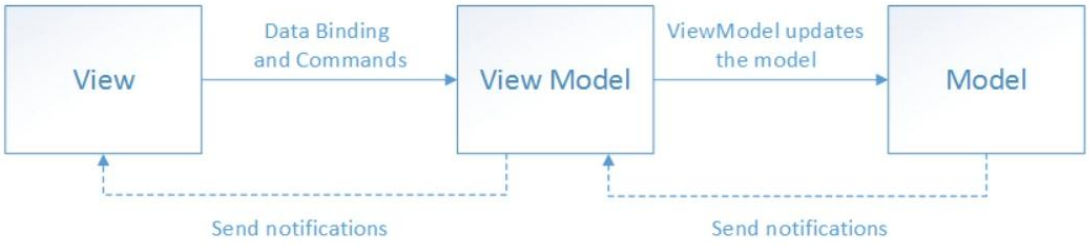


点击“Finish”完成项目创建。

1.2 什么是 MVVM

MVVM 是 Model-View-ViewModel 的简写。它本质上就是 MVC 的改进版。MVVM 模式有助于将应用程序的业务和表示逻辑与用户界面清晰分离。 保持应用程序逻辑和 UI 之间的清晰分离有助于解决许多开发问题，并使应用程序更易于测试、维护和演变。 它还可以显著提高代码重用机会，并允许开发人员和 UI 设计人员在开发应用各自的部分时更轻松地进行协作。

大致的结构如下图所示。



MVVM 模式和 MVC 模式一样，主要目的是分离视图（View）和模型（Model），有几大优点

1. 低耦合。视图（View）可以独立于 Model 变化和修改，一个 ViewModel 可以绑定到不同的"View"上，当 View 变化的时候 Model 可以不变，当 Model 变化的时候 View 也可以不变。

2. 可重用性。可以把一些视图逻辑放在一个 ViewModel 里面，让很多 view 重用这段视图逻辑。

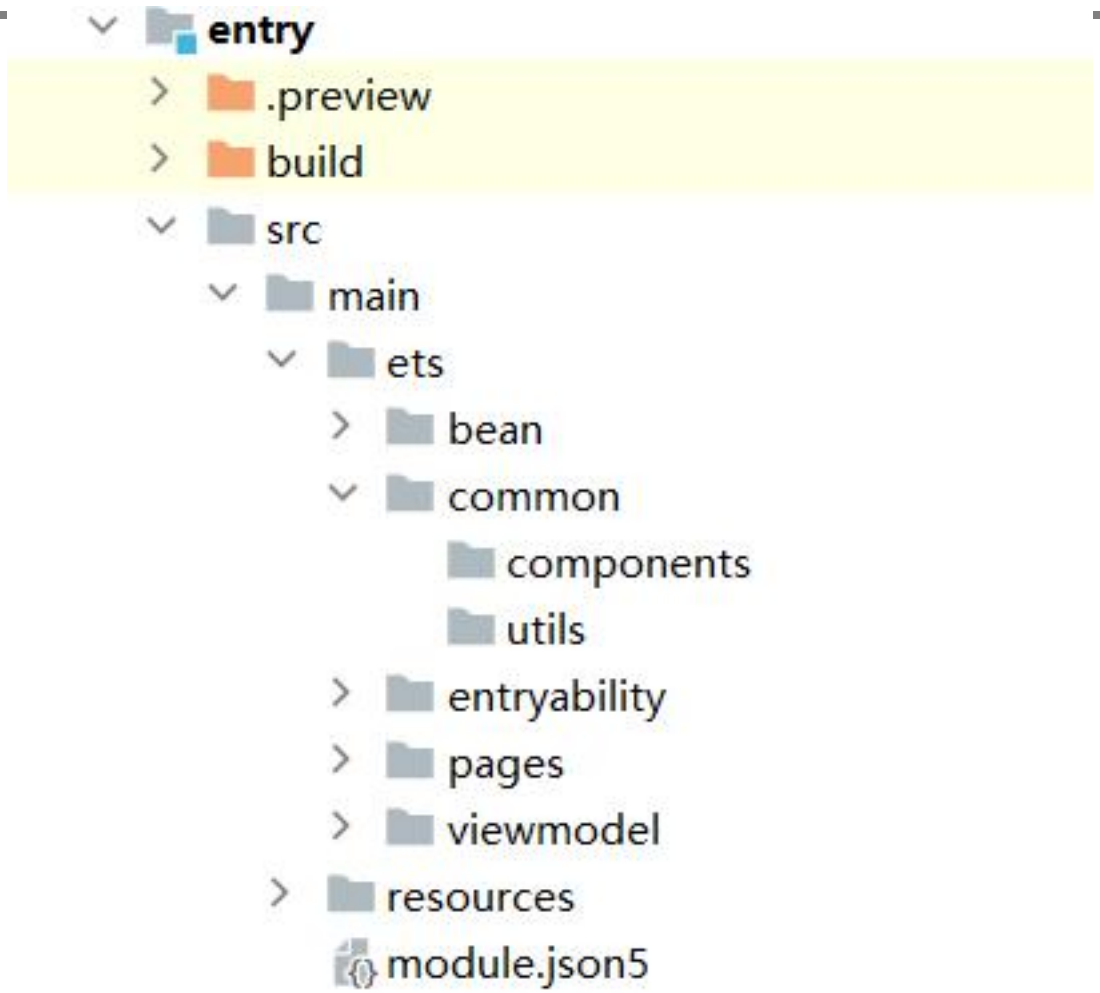
3. 独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计，使用 Expression Blend 可以很容易设计界面并生成 xaml 代码。

4. 可测试。界面素来是比较难于测试的，测试可以针对 ViewModel 来写。 [1]

使用 MVVM 来开发用户控件。由于用户控件在大部分情况下不涉及到数据的持久化，所以如果将 M 纯粹理解为 DomainModel 的话，使用 MVVM 模式来进行自定义**控件开发**实际上可以省略掉 M，变成了 VVM

1.3 创建工程目录结构

如下图所示，组件项目工程目录结构。

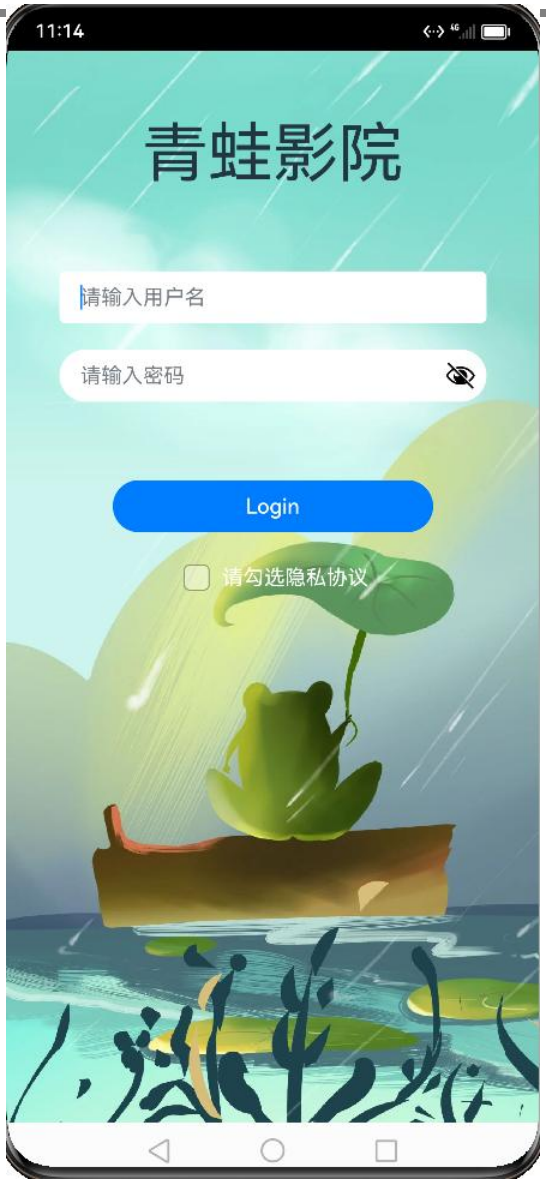


工程目录介绍：

- bean，创建一些 model 类，用于对页面中的数据进行结构化封装。
- common，放一些通用的内容，比如抽离的公共组件、工具类等。
- pages，新建项目就有的，用于保存我们的所有 pages，我们也可以将 page 按模块进行划分。
- viewmodel，用于存放我们的 ViewModel 类。

2 对原型图页面进行布局分解

登录页面的最终效果如下图所示：



（1）页面整体从上至下进行布局，并且在垂直方向上所有内容整体居中，我们可以使用 Column 布局，并为布局容器添加背景图片效果。

（2）“青蛙影院”标题使用 Text 文本组件呈现，并为其设置上下的外边距，使其距离容器顶部和下方的输入框都有一定的间距。

（3）“用户输入框”和“密码输入框”使用 TextInput 组件呈现。登录按钮使用 Button 组件呈现，当登录按钮单击时需要判断用户输入信息是否有误，正确则

跳转到青蛙影院主界面，不正确则提示“用户名或密码错误”。

（4）“勾选隐私协议”部分包含了多选框按钮和文本，可以使用 Row 容器包裹，使两个组件在一行显示。

3 登录模块实现

3.1 登录页面布局实现

我们将 pages 目录下的 Index.ets 修改为 Login.ets，作为我们程序的入口页面。接下来先在该 Login.ets 中来构建登录页面的页面结构，如下所示。

```
@Entry
@Component
struct Login {

  build() {
    Column() {
      //标题
      Text("青蛙影院")

      //用户名输入框
      TextInput({
        placeholder: "请输入用户名"
      })

      //密码输入框
      TextInput({
        placeholder: "请输入密码"
      })


      //登录按钮
      Button("Login")
```

```
      //勾选隐私协议
      Row() {
        Checkbox()
        Text("请勾选隐私协议").fontColor(Color.White)
      }

    }.width("100%")
    .height("100%")
    .backgroundImage($r('app.media.frog'))
    .backgroundImageSize(ImageSize.Cover)
  }
}
```

接下来我们需要对页面进行优化。

①首先是标题部分。我们给标题设置一些间距，让其更加的美观。

```
//标题
      Text("青蛙影院")
        .fontSize(50)
        .margin({
          top: 50,
          bottom: 50
        })
```

②然后是输入账号和密码的输入框。我们发现它们的布局基本一样，这时候我们可以考虑抽离公共样式，公共样式的代码如下所示（该部分代码可放在 Login.ets 所有代码的最上方）。

```
@Styles function myInput(){
  .width("80%")
  .height(40)
  .margin({
    top: 10,
```

```
    bottom: 10
  })
  .backgroundColor(Color.White)
}
```

该公共样式如何调用呢？只需在文本框和密码框中通过.myInput()的形式调用即可。如下所示：

```
//用户名输入框
      TextInput({
        placeholder: "请输入用户名"
      })
      .myInput()

      //密码输入框
      TextInput({
        placeholder: "请输入密码"
      })
      .myInput()
```

③接下来是按钮，基本结构如下。

```
Button("Login")
        .width("60%")
        .height(40)
        .onClick((event?: ClickEvent) => {


          })
```

④隐私协议部分，采用 Row 布局，基本结构如下。

```
//勾选隐私协议
Row({space: 5}) {
  Checkbox()
  Text("请勾选隐私协议").fontColor(Color.White)
}.margin({
```

```
  top: 20
})
```

3.2 属性事件绑定及规则校验

优化完登录页的整体结构布局后，我们来编写登录页的校验逻辑。校验的大概逻辑如下，在用户点击登录按钮时触发校验规则。

规则 1：获取用户是否勾选隐私协议。如果没有勾选，提示用户需要勾选隐私协议，程序到此结束。如果勾选了，再进行下一步校验。

规则 2：校验用户名和密码。首先我们需要获取用户名和密码，这个校验我们实际开发中一般通过网络进行校验，这里我们可以采用本地模拟校验的方式进行。

那么具体如何实现呢？

①首先，我们需要在项目中定义一些@State 变量，用来接收我们的数据，比如输入的用户名、密码、是否勾选隐私协议。

```
@State username: string = ""
@State password: string = ""
@State isSelect: boolean = false
```

②然后对 TextInput 绑定 onChange 事件，用于获取输入的内容。

```
//用户名输入框
    TextInput({
      placeholder: "请输入用户名"
    })
      .myInput()
      .onChange((value: string) => {
        this.username = value
      })
      .borderRadius(5)
    //密码输入框
```

```
      TextInput({
        placeholder: "请输入密码"
      })

        .myInput()
        .onChange((value: string) => {
          this.password = value
        })
        .type(InputType.Password)
```

③获取是否勾选用户隐私协议。

```
//勾选隐私协议
      Row({
        space: 5
      }) {
        Checkbox()
          .select(this.isSelect)
          .onChange((value: boolean) => {
            this.isSelect = value
          })
        Text("请勾选隐私协议").fontColor(Color.White)
      }.margin({
        top: 20
      })
```

④我们还需要对我们的按钮设置点击事件，用于触发这些校验规则。由于当用户输入正确的信息后页面需要进行跳转，所以这里需要引入路由模块。引入方式如下：

```
import router from '@ohos.router'
```

按钮点击的事件逻辑如下：

```
//登录按钮
      Button("Login")
```

```
      .width("60%")
      .height(40)
      .onClick((event?: ClickEvent) => {
        //获取用户名和密码
        //校验数据库
        //是否勾选用户协议

        if (!this.isSelect) {
          //弹出对话框进行用户提醒
          AlertDialog.show({
            title: "提示",
            message: "请勾选隐私协议",
            autoCancel: true
          })
          return
        }
        if (this.username == "admin" && this.password == "123456") {
          //跳转
          router.pushUrl({
            url: "pages/LoadingPage"
          })
        } else {
          AlertDialog.show({
            title: "提示",
            message: "用户名或密码错误",
            autoCancel: true
          })
        }

      })
      .margin({
        top: 50
      })
```

到此为止，我们登录页已全部完成

3 登录页的完整代码

此时登录页的完整代码如下所示：

```
import router from '@ohos.router'

@Styles function myInput(){
  .width("80%")
  .height(40)
  .margin({
    top: 10,
    bottom: 10
  })
  .backgroundColor(Color.White)
}

@Entry
@Component
struct Login {
  @State username: string = ""
  @State password: string = ""
  @State isSelect: boolean = false

  build() {
    Column() {
      //标题
      Text("青蛙影院")
        .fontSize(50)
        .margin({
          top: 50,
          bottom: 50
        })

      //用户名输入框
      TextInput({
        placeholder: "请输入用户名"
```

```
  })
    .myInput()
    .onChange((value: string) => {
      this.username = value
    })
    .borderRadius(5)
  //密码输入框
  TextInput({
    placeholder: "请输入密码"
  })

    .myInput()
    .onChange((value: string) => {
      this.password = value
    })
    .type(InputType.Password)


  //登录按钮
  Button("Login")
    .width("60%")
    .height(40)
    .onClick((event?: ClickEvent) => {
      //获取用户名和密码
      //校验数据库
      //是否勾选用户协议

      if (!this.isSelect) {
        //弹出对话框进行用户提醒
        AlertDialog.show({
          title: "提示",
          message: "请勾选隐私协议",
          autoCancel: true
        })
```

```
            return
          }
          if (this.username == "admin" && this.password == "123456") {
            //跳转
            router.pushUrl({
              url: "pages/LoadingPage"
            })
          } else {
            AlertDialog.show({
              title: "提示",
              message: "用户名或密码错误",
              autoCancel: true
            })
          }

        })
        .margin({
          top: 50
        })

      //勾选隐私协议
      Row({
        space: 5
      }) {
        Checkbox()
          .select(this.isSelect)
          .onChange((value: boolean) => {
            this.isSelect = value
          })
        Text("请勾选隐私协议").fontColor(Color.White)
      }.margin({
        top: 20
      })

    }.width("100%")
```

```
    .height("100%")
    .backgroundImage($r('app.media.frog'))
    .backgroundImageSize(ImageSize.Cover)
  }
}
```