HUAWEI

< HDC.Together >

HUAWEI DEVELOPER CONFERENCE 2021

图库应用设计开发实践

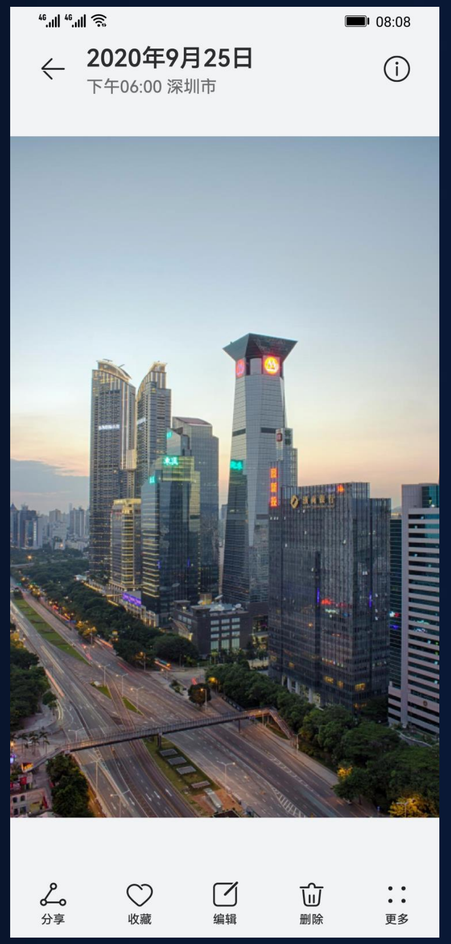
1 UI组件设计开发实践

2 组件解耦实践

图库应用介绍

照片列表

相册列表



大图列表

应用数据加载显示模型

列表高性能加载

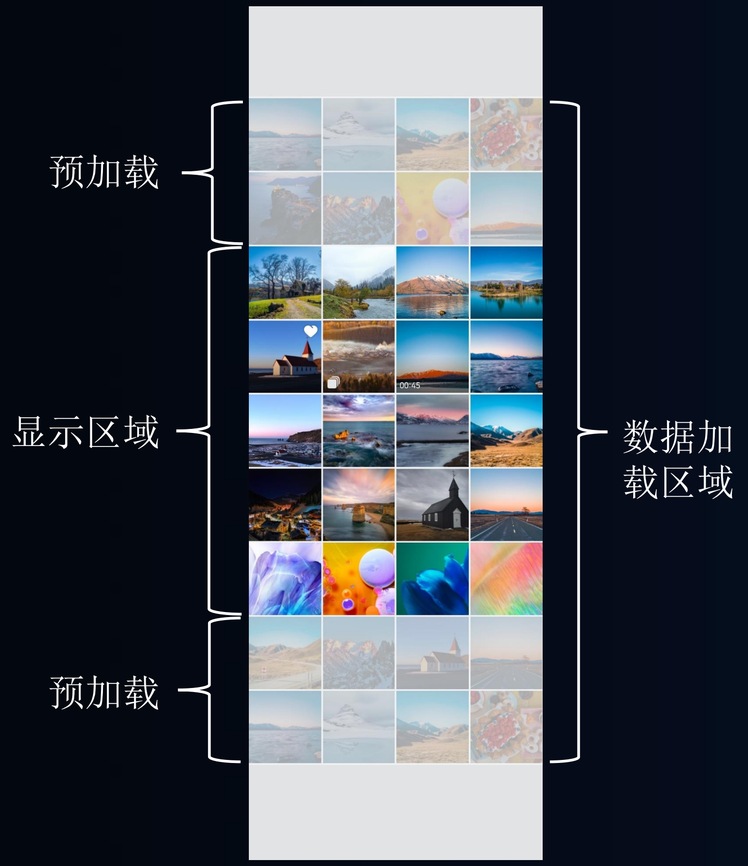
缩略图缓存

框架复用

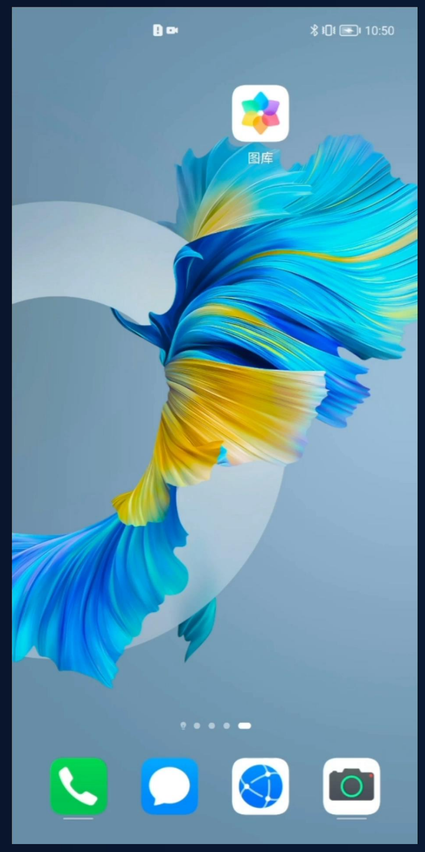
应用数据加载显示模型

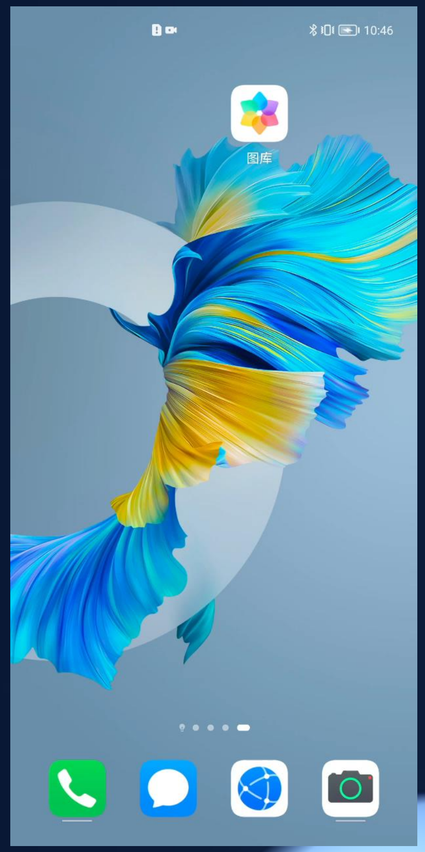

```
@Component
export struct GridPhoto {
  @State data: ItemBean[] = []

  private aboutToAppear(): void {
      this.prepareData()
  }

  build() {
    Flex({
      direction: FlexDirection.Column,
      alignItems: ItemAlign.Start
    }) {
      Stack() {
        Grid() {
          ForEach(this.data, (itemBean) => {
            GridItem() {
              ImageComponent({
                item: itemBean
              })
            }
          }, item => JSON.stringify(item))
        }.columnsTemplate(TemplateModel.COLUMN_FOUR_PARAM)
        .columnsGap(UiConstant.PHOTO_SLOT_GAP)
        .rowsGap(UiConstant.PHOTO_SLOT_GAP)
      }
    }
  }

  async prepareData() {...}
}
```

应用数据按需加载显示模型




```
@Component
export struct GridPhoto {
  dataSource: DataSource

  private aboutToAppear(): void {
    this.dataSource = DataSourceFactory.getDataSource()
  }

  build() {
    Flex({
      direction: FlexDirection.Column,
      alignItems: ItemAlign.Start
    }) {
      Stack() {
        Grid() {
          LazyForEach(this.dataSource, (itemBean) => {
            GridItem() {
              ImageComponent({
                item: itemBean
              })
            }
          }, item => JSON.stringify(item))
        }.columnsTemplate(TemplateModel.COLUMN_FOUR_PARAM)
        .columnsGap(UiConstant.PHOTO_SLOT_GAP)
        .rowsGap(UiConstant.PHOTO_SLOT_GAP)
        .cachedCount(UiConstant.DEFAULT_CACHE_COUNT)
      }
    }
  }

  async prepareData() {...}
}
```





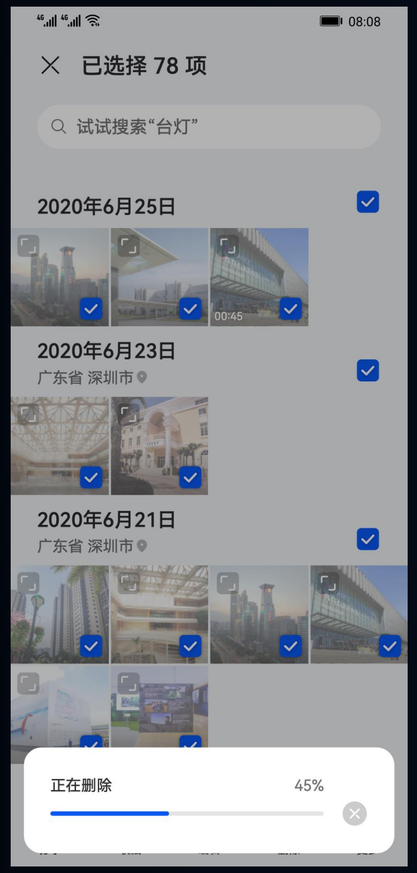
应用数据按需加载显示模型

```
@Component
export struct GridPhoto {
  dataSource: DataSource

  private aboutToAppear(): void {
    this.dataSource = DataSourceFactory.getDataSource()
  }

  build() {
    Row() {
      Stack() {
        Swiper() {
          LazyForEach(this.dataSource, (itemBean) => {
            ImageComponent({
              item: itemBean
            })
          }, item => JSON.stringify(item))
        }
      }
    }
  }

  async prepareData() {...}
}

async prepareData() {...}
}
```

1 UI组件设计开发实践

2 组件解耦实践

组件解耦实践 @Link



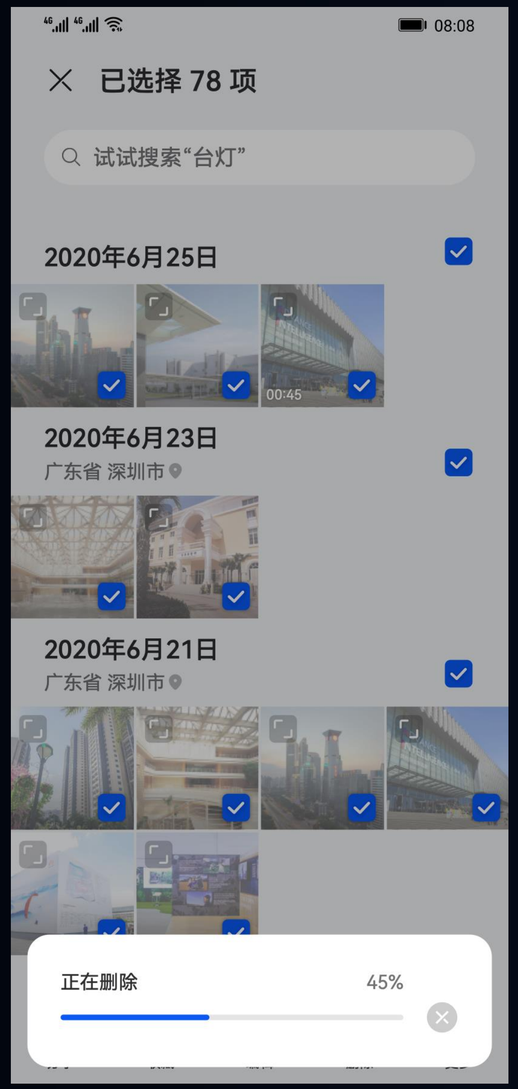

```
@Component
export struct AlertCustomDialog {
    @State isShow: boolean = true

    build() {
        Row() {
            Stack() {
                Row().width(ScreenUtil.FULL_WIDTH)
                    .height(ScreenUtil.FULL_HEIGHT)
                    .opacity(UiConstant.DEFAULT_OPACITY)
                    .backgroundColor($r('app.color.black'))

                Row() {
                    // 自定义DialogView
                    ProgressDialogView({isShow: $isShow})
                }
            }.width(ScreenUtil.FULL_WIDTH).height(ScreenUtil.FULL_HEIGHT)
            .visibility(this.isShow ? Visibility.Visible : Visibility.None)
        }
    }
}
```

```
@Component
export struct ProgressDialogView {
    @Link isShow: boolean
    @Consume deleteProgress: number

    build() {
        Column() {
            Row() {
                Text($r('app.string.photo_deleting')).flexGrow(1)
                Text(`${this.deleteProgress}%`)
            }.width(ScreenUtil.FULL_WIDTH)

            Row() {
                Progress({value:MathConstant.ZERO_PERCENTAGE,
                    total:MathConstant.FULL_PERCENTAGE,
                    style:ProgressStyle.Linear})
                    .value(this.deleteProgress)

                Image($r('app.media.ic_progress_cancel')).onClick(()=> {
                    this.isShow = false
                })
            }
        }
        .width(ScreenUtil.DIALOG_WIDTH)
    }
}
```

删除进度Component

对话框Component

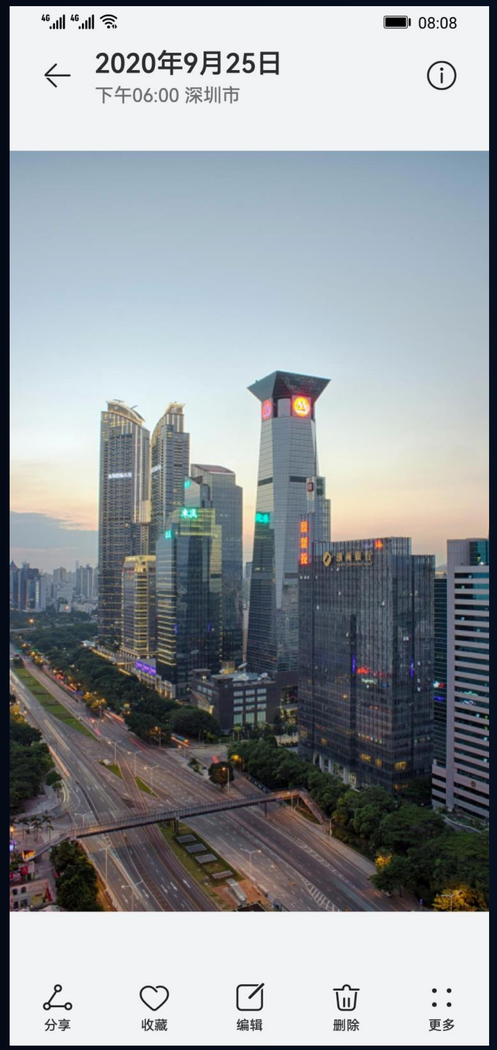
组件解耦实践 @Watch




```
@Component
export struct AlertCustomDialog {
    @State @Watch('onDialogVisibilityUpdated') isShow: boolean = true
    @State viewVisible: boolean = false

    onDialogVisibilityUpdated() : void {
        if (this.isShow) {
            this.enterAnimation()
        } else {
            this.leaveAnimation()
        }
    }

    build() {
        Row() {
            Stack() {
                Row().width(ScreenUtil.FULL_WIDTH).height(ScreenUtil.FULL_HEIGHT)
                    .opacity(UiConstant.DEFAULT_OPACITY)
                    .backgroundColor($r('app.color.black'))

                Row() {
                    // 自定义DialogView
                    ProgressDialogView({isShow: $isShow})
                }
            }.width(ScreenUtil.FULL_WIDTH).height(ScreenUtil.FULL_HEIGHT)
            .visibility(this.viewVisible ? Visibility.Visible : Visibility.None)
        }
    }

    leaveAnimation() {
        animateTo({duration: 220,
            curve: Curves.cubicBezier(UiConstant.CUBIC_BEZIER_LEAVE_P1_X,
                UiConstant.CUBIC_BEZIER_LEAVE_P1_Y,
                UiConstant.CUBIC_BEZIER_LEAVE_P2_X,
                UiConstant.CUBIC_BEZIER_LEAVE_P2_Y)},
            () => { this.viewVisible = false })
    }
```

```
@Component
export struct ProgressDialogView {
    @Link isShow: boolean
    @Consume deleteProgress: number

    build() {
        Column() {
            Row() {
                Text($r('app.string.photo_deleting')).flexGrow(1)
                Text(`${this.deleteProgress}%`)
            }.width(ScreenUtil.FULL_WIDTH)

            Row() {
                Progress({value:MathConstant.ZERO_PERCENTAGE,
                    total:MathConstant.FULL_PERCENTAGE,
                    style:ProgressStyle.Linear})
                    .value(this.deleteProgress)

                Image($r('app.media.ic_progress_cancel')).onClick(()=> {
                    this.isShow = false
                })
            }
        }
        .width(ScreenUtil.DIALOG_WIDTH)
    }
}
```

组件解耦实践 @Provide @Consume



地点Component

导航Component

大图Component

组件解耦实践 @Provide @Consume

地点Component

导航Component

大图Component

```
@Component
struct PhotoBrowser {
    @Provide("locationTitle") location: string = ''

    updateActionBar() {
        this.location = DateUtil.formatAddress(this.address)
    }

    build() {
        Stack({alignContent:Alignment.TopStart}) {
            PhotoBrowserBg()
            BigPhotoSwiper({
                dataSource: this.dataSource,
                onPhotoChanged: this.onPhotoChanged.bind(this),
            })
            PhotoActionBar({
                onMenuClicked: this.onMenuClicked,
                leftBlank: $leftBlank
            })
        }
    }
}
```

```
@Component
export struct DetailTitle {
    @Consume("locationTitle") subTitle: string

    build() {
        Row() {
            Column() {
                Text(this.title)
                    .fontSize(ActionBarProp.TITLE_TEXT_SIZE)
                    .fontColor(ActionBarProp.NORMAL_TEXT_COLOR)
                    .fontFamily(ActionBarProp.MEDIUM_FONT)
                    .fontWeight(ActionBarProp.TITLE_FONT_WEIGHT)
                Text(this.subTitle)
                    .fontSize(ActionBarProp.SUBTITLE_TEXT_SIZE)
                    .fontFamily(ActionBarProp.REGULAR_FONT)
                    .fontColor(ActionBarProp.SUBTITLE_TEXT_COLOR)
            }
            .alignItems(HorizontalAlign.Start)
        }
        .alignItems(VerticalAlign.Center)
    }
}
```

组件解耦实践 @StorageLink

组件解耦实践 @ StorageLink

```
@Component
struct PhotoBrowser {
    @StorageLink('SLOT_PAGE_INDEX') slotPageIndex: number
        = Constants.INVALID

    private onBackPress() {
        // currentIndex : 大图左右滑动最终停留位置
        this.slotPageIndex = this.currentIndex
        router.back({
            params: {
                index: this.currentIndex
            }
        })
    }
}
```

```
15  AppStorage.SetOrCreate('SLOT_PAGE_INDEX', Constants.INVALID)
16  @Component
17  export struct TimelinePage {
18      @StorageLink('SLOT_PAGE_INDEX') @Watch("onIndexChange")
19          slotIndex: number = Constants.INVALID
20
21      onIndexChange(index: number) {
22          if (this.slotIndex != Constants.INVALID) {
23              this.scroller.scrollToIndex(
24                  this.dataSource.getPositionByIndex(this.slotIndex))
25          }
26      }
27  }
```

总结

1、ArkUI组件式开发使得模块更加内聚、更易于扩展复用

2、通过框架装饰器大大降低组件之间的耦合

扫码参加1024程序员节

〈解锁HarmonyOS核心技能，赢取限量好礼〉

开发者训练营

Codelabs 挑战赛

HarmonyOS技术征文

HarmonyOS开发者创新大赛

扫码了解1024更多信息

报名参加HarmonyOS开发者创新大赛

谢谢

欢迎访问HarmonyOS开发者官网

欢迎关注HarmonyOS开发者微信公众号