HUAWEI

玩转服务卡片

华为云 一切皆服务

前言

- ExtensionAbility组件是基于特定场景提供的应用组件，以便满足更多的使用场景。
- FormExtensionAbility： FORM类型的ExtensionAbility组件，用于提供服务卡片场景相关能力。
- 服务卡片是一种界面展示形式，可以将应用的重要信息或操作前置到卡片，以达到服务直达、减少体验层级的目的。
- 本章将带领开发者们认识服务卡片，了解服务卡片的核心概念与实现原理，学习基于ArkTS UI的服务卡片开发流程。

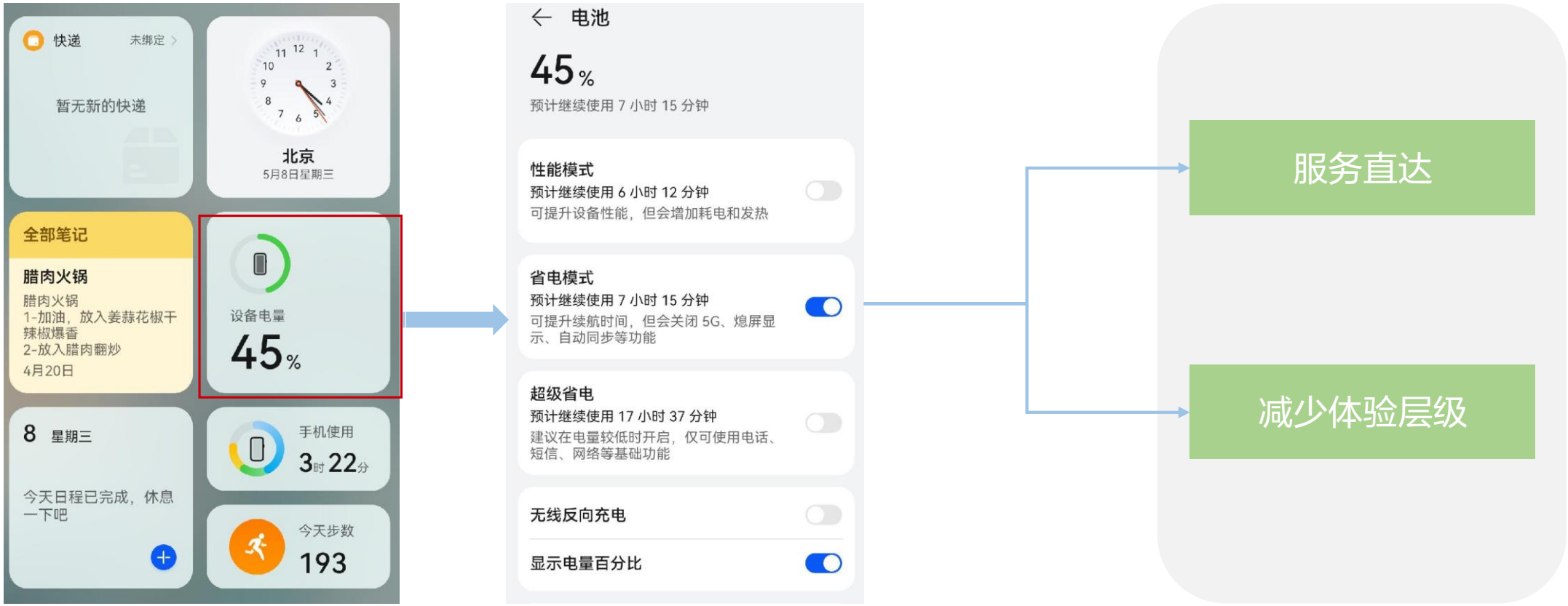
课程目标

- 学完本课程后，您将能够：
  - 了解服务卡片的核心概念；
  - 了解ArkTS卡片的相关原理；
  - 掌握基于ArkTS UI的卡片开发流程。

目录

什么是服务卡片？

- 服务卡片（以下简称“卡片”）是一种界面展示形式，可以将应用的重要信息或操作前置到卡片，以达到服务直达、减少体验层级的目的。



服务卡片的定义及基本概念

卡片定义

1. 作为一种界面展示形式，可将重要信息或操作前置到卡片，以达到服务直达、减少体验层级的目的。

2. 卡片常用于嵌入其他系统应用中，作为其页面展示的一部分，支持拉起页面、发送消息等基础交互功能。

基本概念

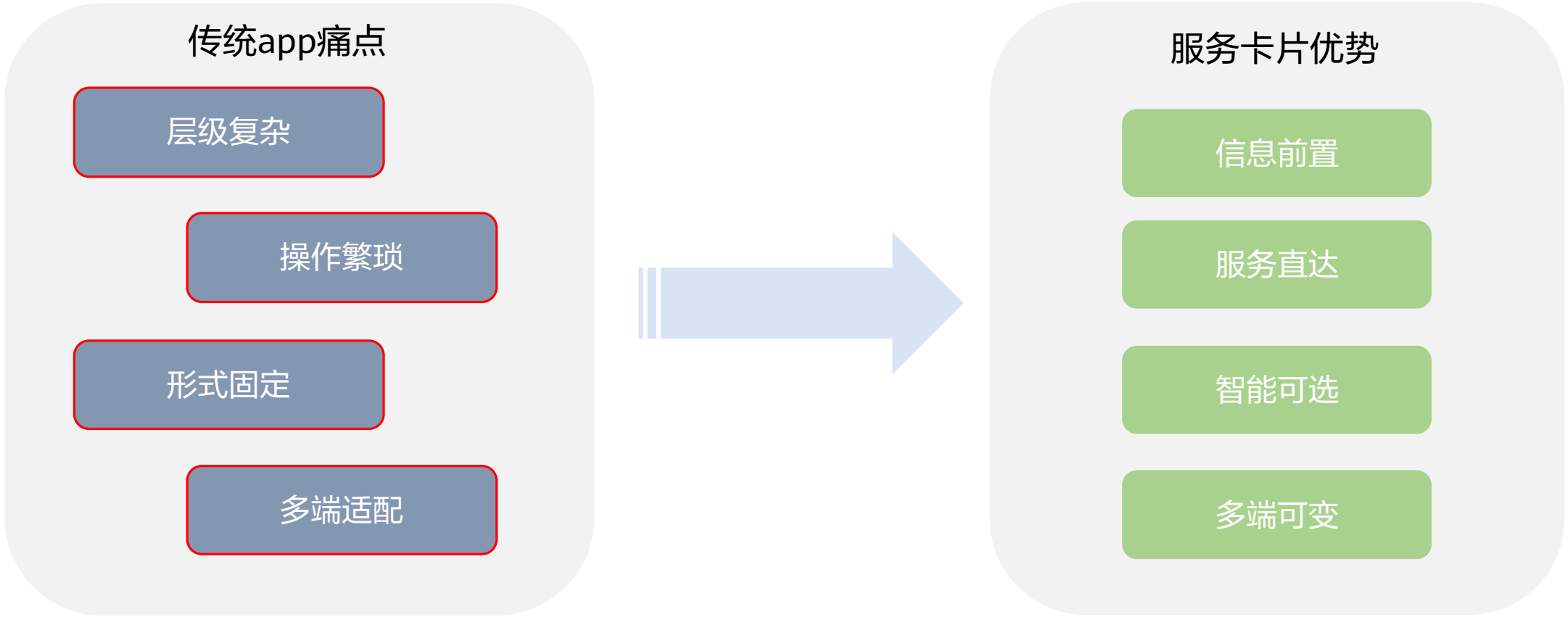
1. **卡片提供方：**
包含卡片的应用，提供卡片的显示内容可以进行交互，如界面刷新、应用跳转等。

2. **卡片使用方：**
如图中的桌面，控制卡片的展示位置。

HUAWEI

服务卡片的优势

- 相较于传统app给用户带来的操作繁琐感，服务卡片无疑带来了更加清爽流畅的体验，其具有的优势如下：



服务卡片的应用场景

- 基于服务卡片的众多优势，让其拥有了多元化的应用场景，给用户带来高效与便捷的极致体验。

HUAWEI

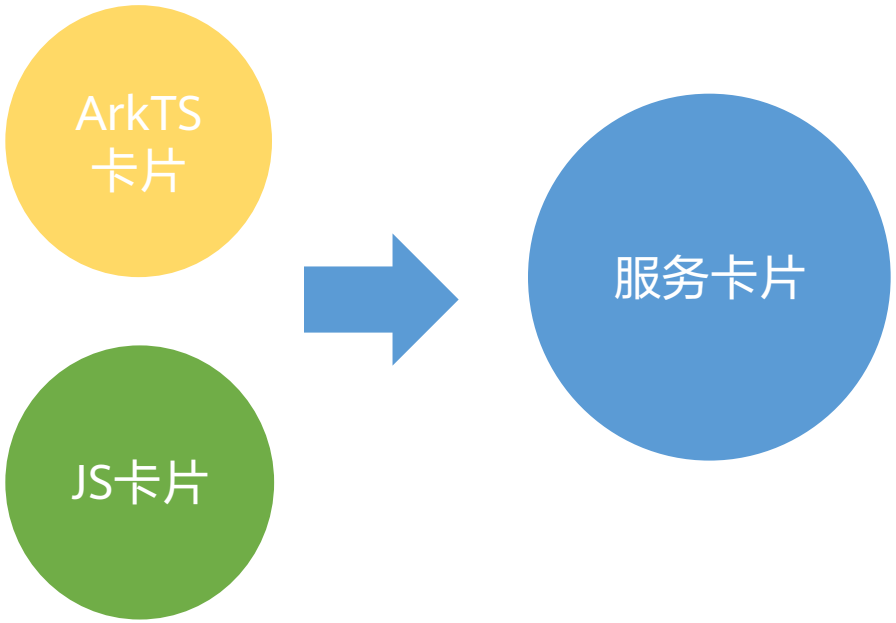
服务卡片的常见使用

- 服务卡片的常见使用步骤如下：

HUAWEI

服务卡片的UI页面开发方式 (1)

- 在Stage模型下，服务卡片UI页面支持通过ArkTS和JS两种语言进行开发：
  - 基于声明式范式ArkTS UI开发的卡片，简称ArkTS卡片。
  - 基于类Web范式JS UI开发的卡片，简称JS卡片。



服务卡片的UI页面开发方式 (2)

- ArkTS卡片与JS卡片具备不同的实现原理及特征，在场景能力上的差异如下表所示：

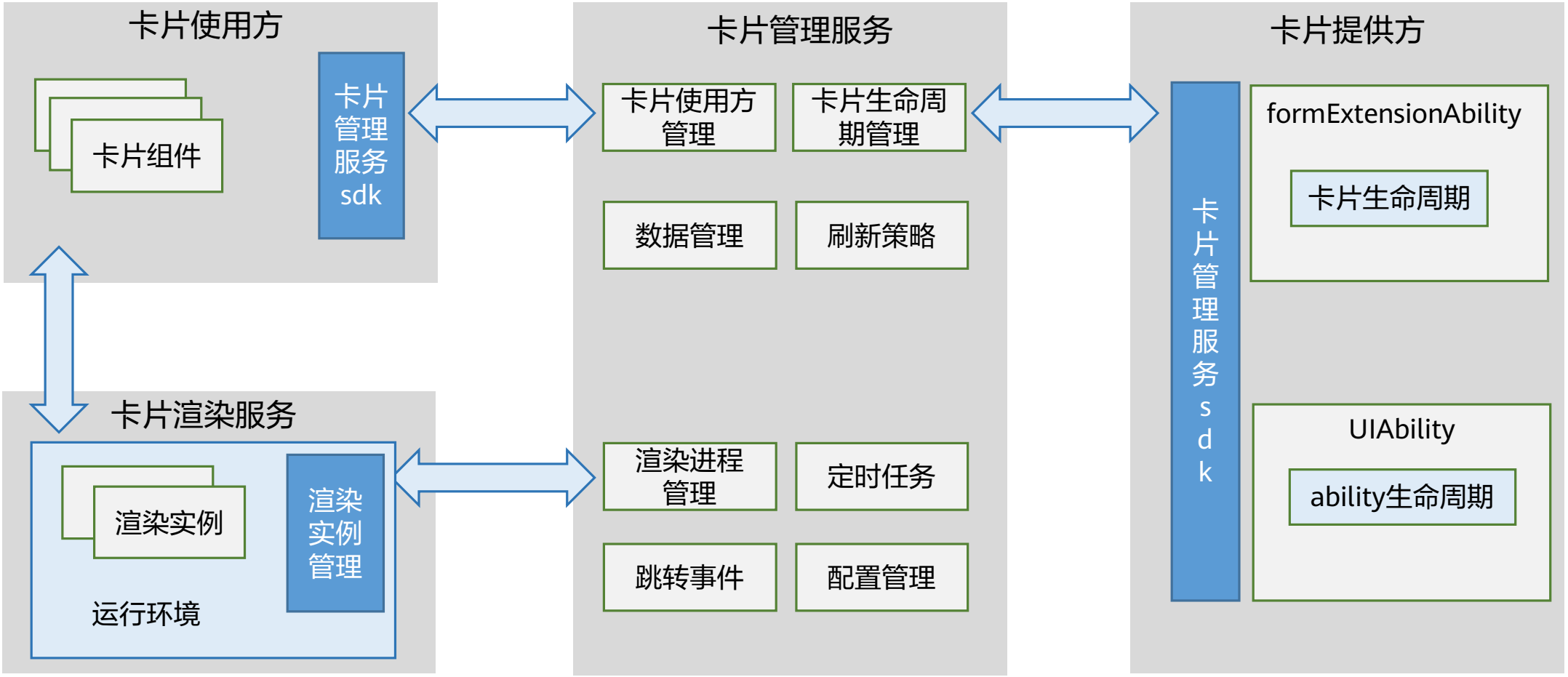
类别	JS卡片	ArkTS卡片（推荐）
开发范式	类web范式	声明式范式
组件能力	支持	支持
布局能力	支持	支持
事件能力	支持	支持
自定义动效	不支持	支持
自定义绘制	不支持	支持
逻辑代码执行（不包含import能力）	不支持	支持

目录

HUAWEI

ArkTS卡片实现原理

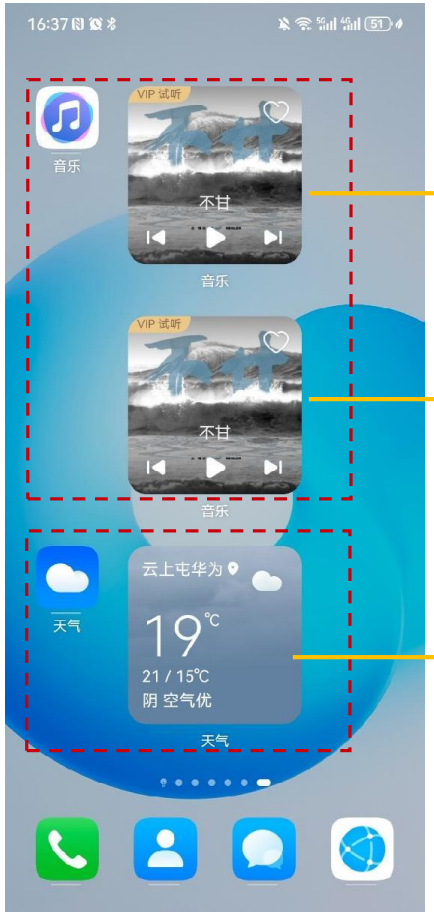
- ArkTS卡片实现原理如下：

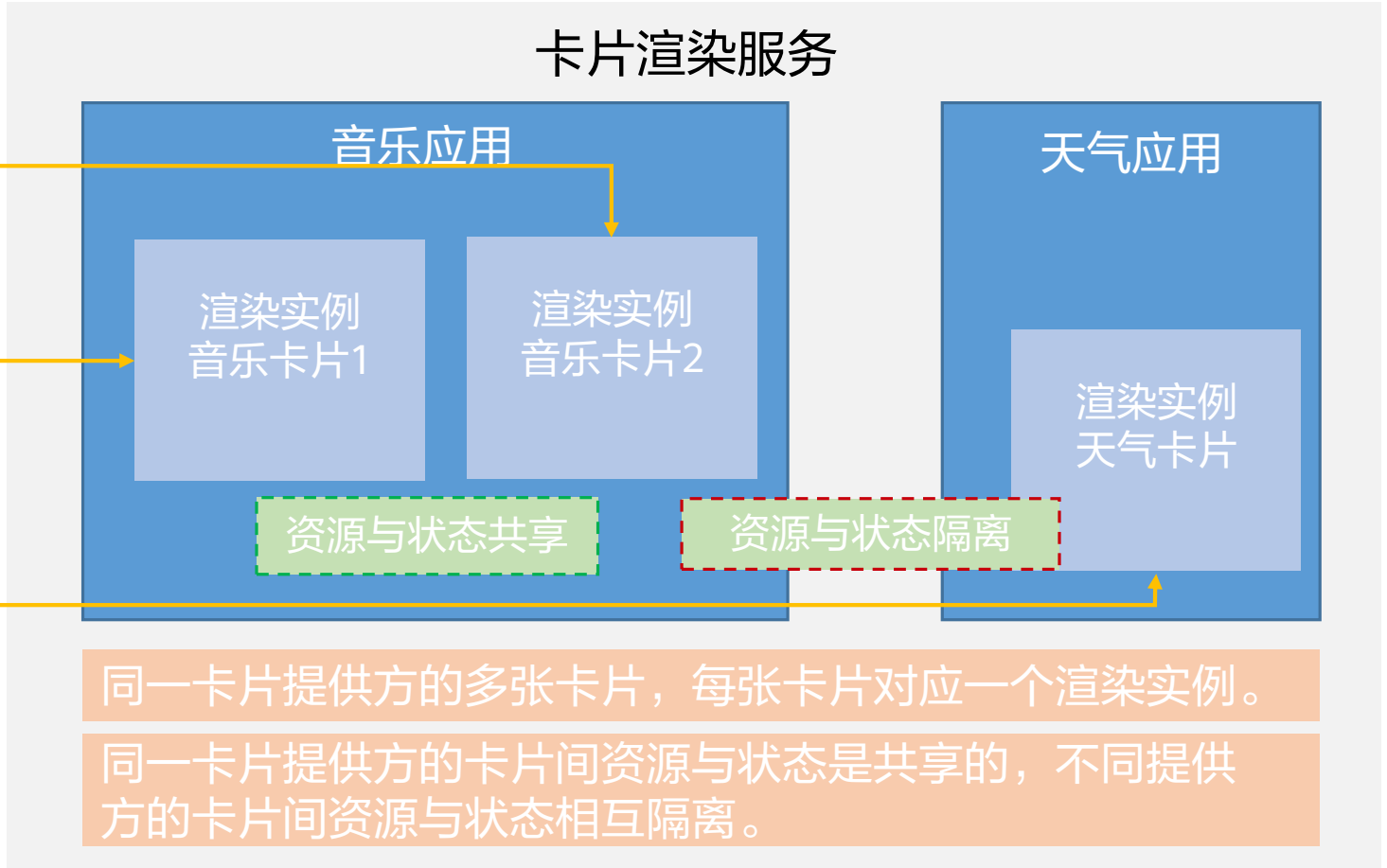


HUAWEI

ArkTS卡片渲染服务运行原理

- ArkTS卡片的渲染服务运行原理如下：



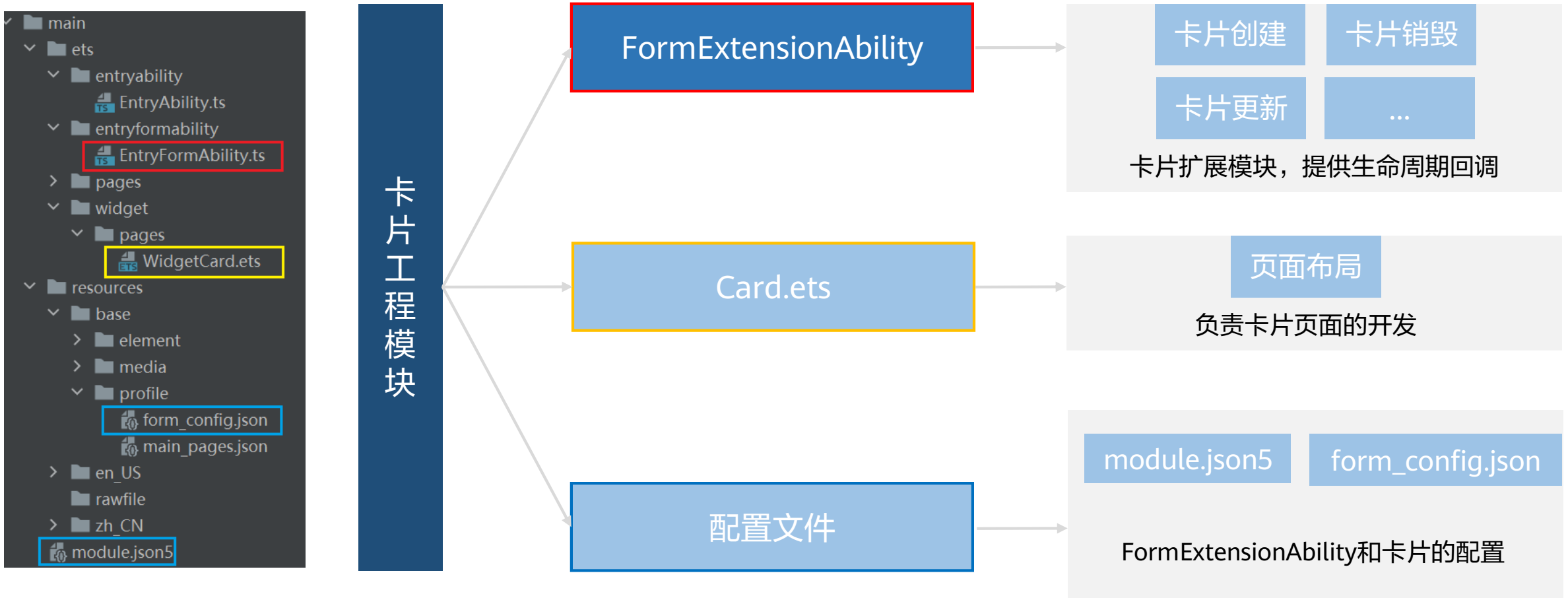


目录

HUAWEI

ArkTS卡片工程模块

- ArkTS卡片的工程模块如下：



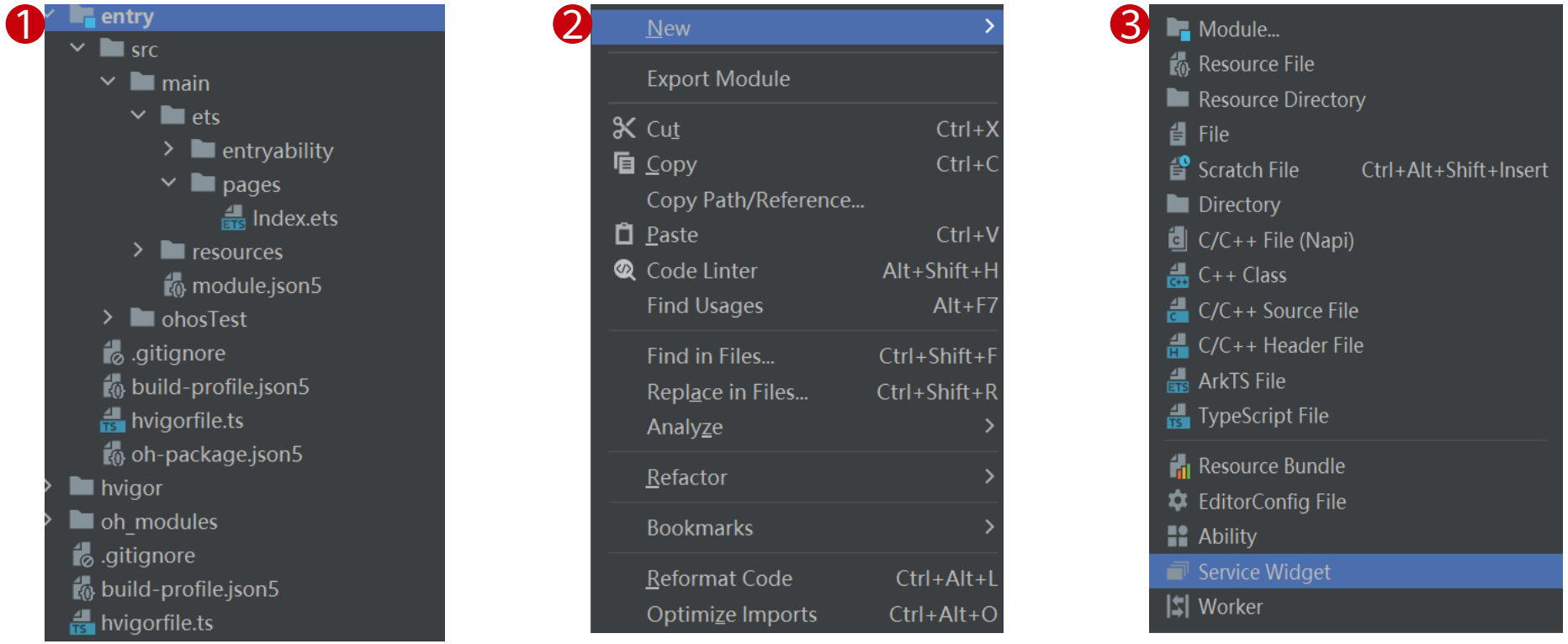
HUAWEI

目录

HUAWEI

ArkTS卡片开发 - 创建卡片 (1)

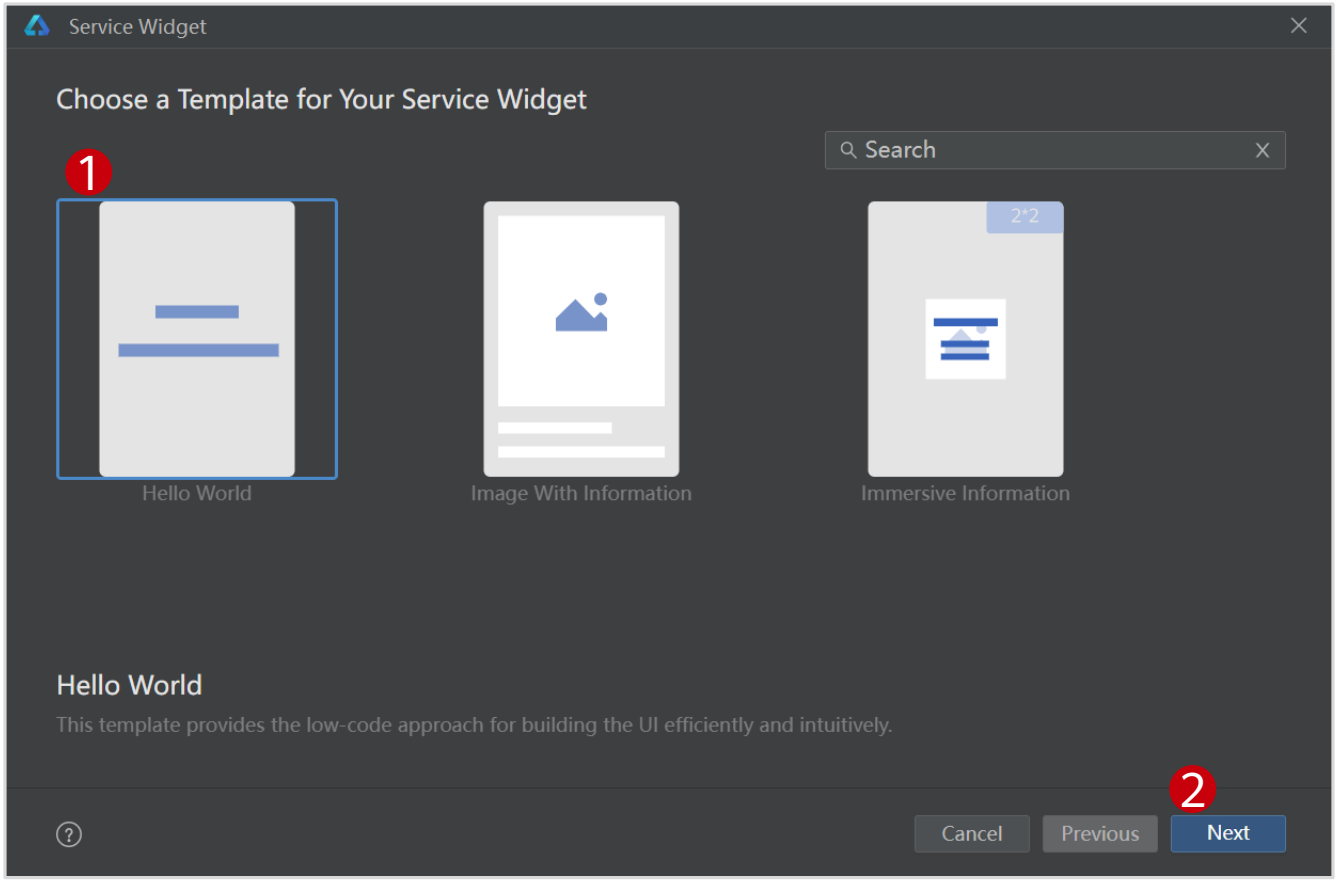
- 在已有的应用工程中，右键entry目录，点击 New > Service Widget。



HUAWEI

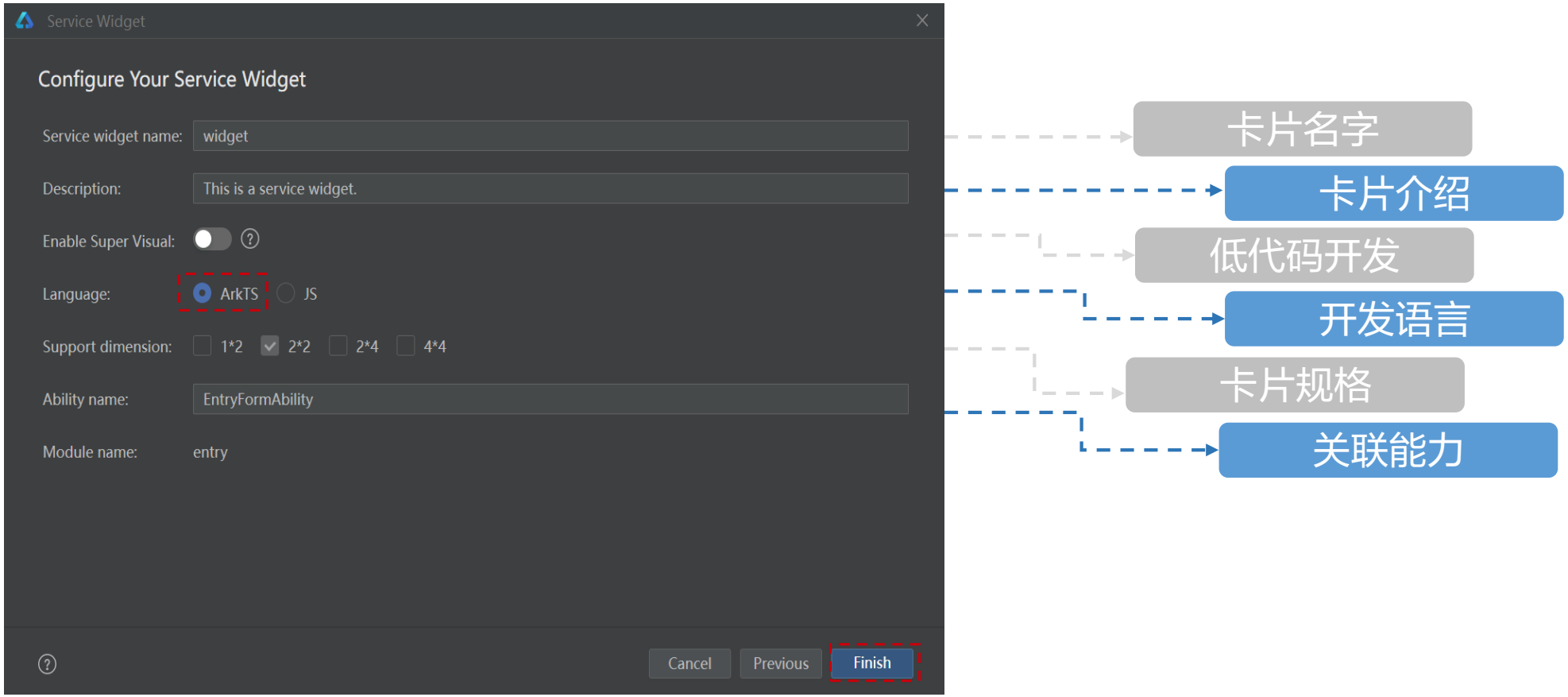
ArkTS卡片开发 - 创建卡片 (2)

- 根据实际业务场景，选择不同的卡片模板，点击“Next” 按钮。



ArkTS卡片开发 - 创建卡片 (3)

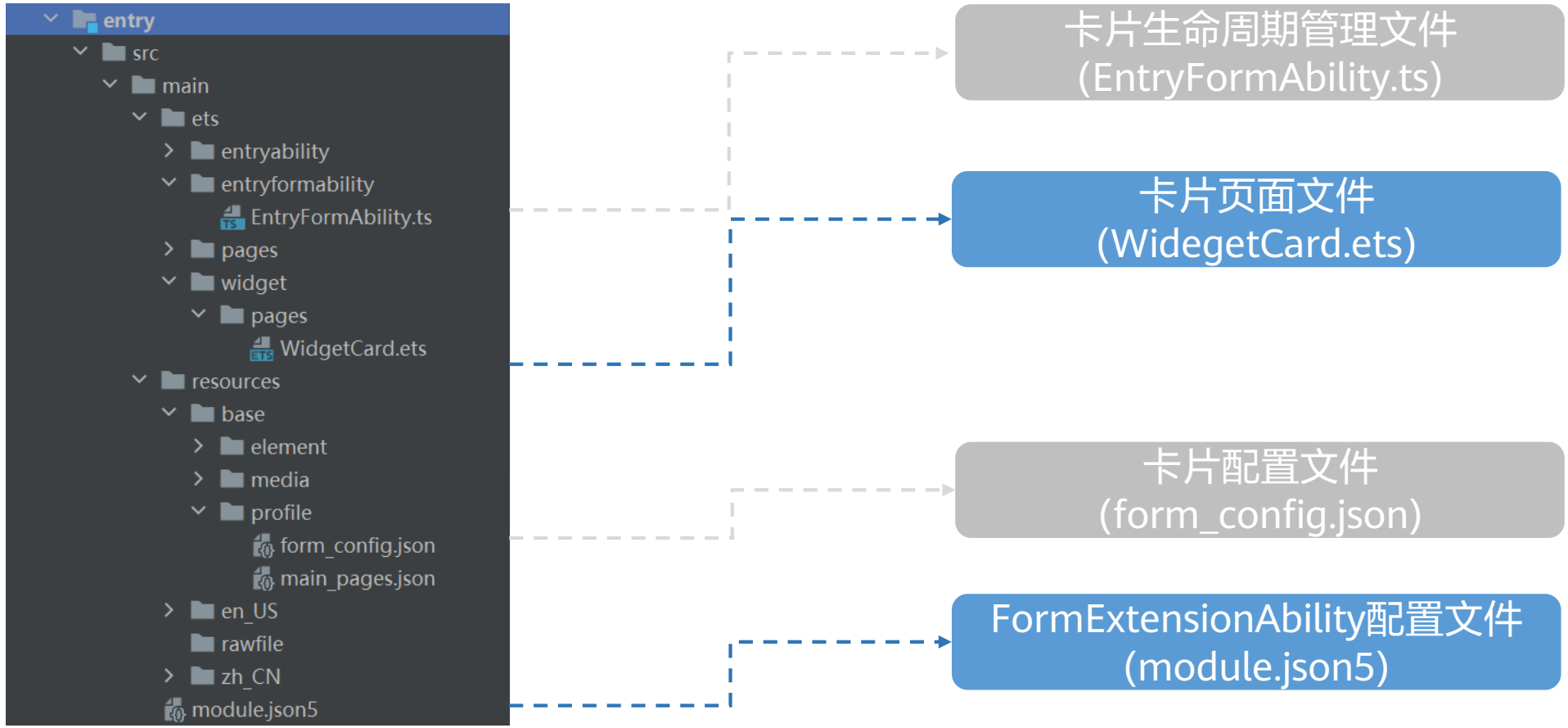
- 配置卡片信息，选择卡片的开发语言，单击“Finish”，即可完成ArkTS卡片创建。



HUAWEI

ArkTS卡片开发 - 工程目录

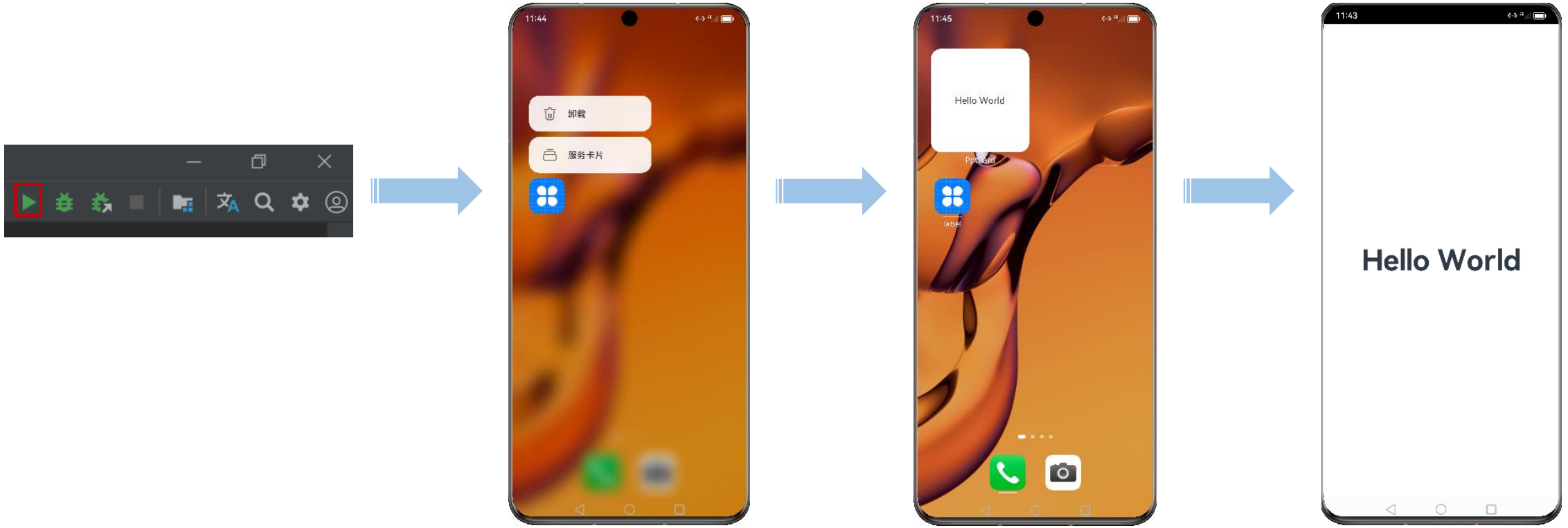
- 卡片创建完成后，生成新的卡片相关文件，完整工程目录如下所示：



HUAWEI

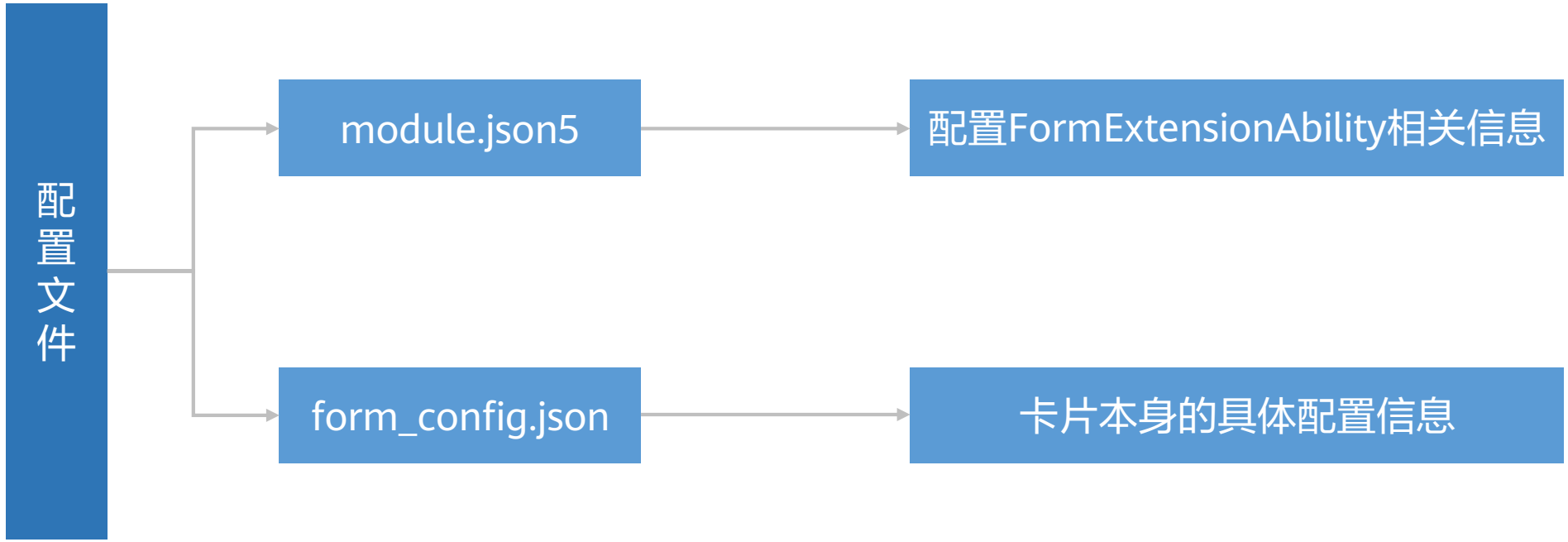
ArkTS卡片开发 - HelloWorld

- 卡片创建完成后，可以点击DevEcoStudio开发工具右上角的启动按钮，运行项目查看效果：



ArkTS卡片开发 - 配置文件

- 卡片相关的配置文件主要包含FormExtensionAbility的配置和卡片的配置两部分。



ArkTS卡片开发 - 配置文件（module.json5 ）

- 在module.json5文件的extensionAbilities 标签下，配置FormExtensionAbility相关的信息。

```
{
  "module": {
    ...
    "extensionAbilities": [
      {
        ...
    "metadata": [
          {
            "name": "ohos.extension.form",          //名称为固定字符串”ohos.extension.form”
            "resource": "$profile:form_config",     //资源为卡片的具体配置信息的索引
                }
            ]
          }
      ]
    }
}
```

此时,form_config.json文件就作为卡片的配置文件

HUAWEI

ArkTS卡片开发 - 配置文件 (form_config.json)

- 当在module.json5文件中，resource指定为$profile:form_config时，form_config.json文件就作为卡片的配置文件，进行卡片的相关配置。

```
{
  "forms": [
    {
      ...
      "colorMode": "auto",
      "isDefault": true,
      "updateEnabled": true, //表示卡片是否支持周期性刷新
      "scheduledUpdateTime": "10:30", //表示卡片的定点刷新时刻，采用24小时制，精确到分钟
      "updateDuration": 1, //表示卡片定时刷新的更新周期，单位为30分钟，两种方式任选其一，同时配置，定时刷新优先生效
      "defaultDimension": "2*2",
      "supportDimensions": [
        "2*2"   ]
          }
     ]
}
```

ArkTS卡片开发 - 生命周期

- 在EntryFormAbility.ts文件中，实现了FormExtensionAbility的生命周期接口，开发者可根据自身业务需求，调用对应的接口。

```
export default class EntryFormAbility extends FormExtensionAbility {
  ① onAddForm(want) {...}
  ② onCastToNormalForm(formId) {...}
  ③ onUpdateForm(formId) {...}
  ④ onChangeFormVisibility(newStatus) {...}
  ⑤ onFormEvent(formId, message) {...}
  ⑥ onRemoveForm(formId) {...}
  ⑦ onConfigurationUpdate(config) {...}
  ⑧ onAcquireFormState(want) {...}
}
```

1. 创建卡片时触发
2. 临时卡片转常态卡片时触发
3. 卡片更新时触发
4. 修改卡片可见性时触发
5. 处理卡片事件时触发
6. 卡片销毁时触发
7. 系统配置更新时触发
8. 查询卡片状态时触发

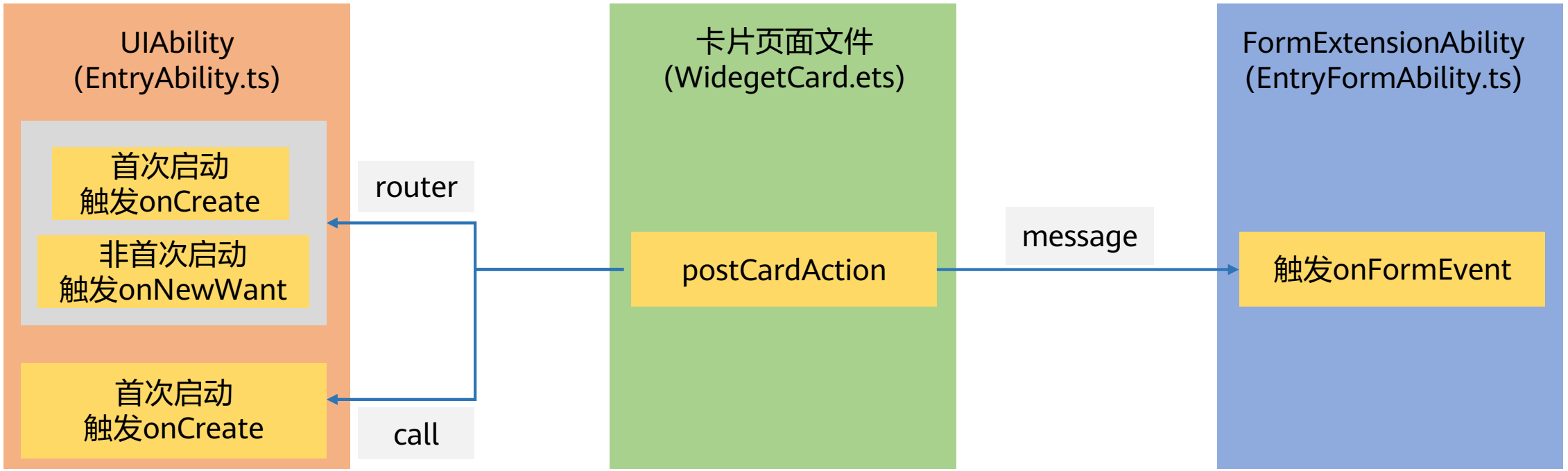
ArkTS卡片开发 - 卡片页面

- 开发者可以使用声明式范式开发ArkTS卡片页面，支持基础组件、容器组件、通用属性、动画等多种页面能力。
- 需要注意卡片场景下的能力差异，例如动效实现旋转动画，与页面能力有限制差别。

名称	参数说明	限制描述
duration	动画播放时长	限制最大的动效播放时长为1秒，当设置大于1秒的时间时，动效时长仍为1秒。
tempo	动画播放速度	卡片中禁止设置此参数，使用默认值1。
delay	动画延迟执行的时长	卡片中禁止设置此参数，使用默认值0。
iterations	动画播放次数	卡片中禁止设置此参数，使用默认值1。

ArkTS卡片开发 - 卡片事件概述

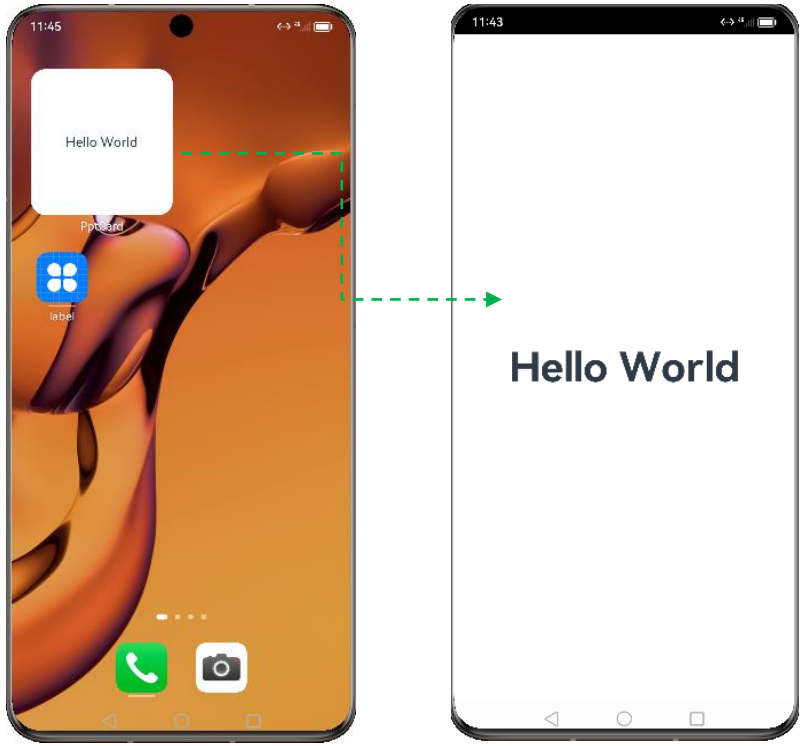
- ArkTS卡片提供了postCardAction()接口，用于卡片内部和提供方应用间的交互，当前支持router、message和call三种类型的事件，仅在卡片中可以调用。



卡片事件 - router (1)

- 使用router事件跳转到指定UIAbility。
  - 在卡片页面文件中调用router事件能够快速拉起卡片提供方应用的指定UIAbility，实现一键直达的效果。

```
//卡片页面文件WidegetCard.ets
...
build() {
    Column() {
      Button('功能A')
      .onClick(() => {
          ...
          postCardAction(this, {
            'action': 'router',    // router、message、call
            'abilityName': 'EntryAbility', // 只能跳转到当前应用下的UIAbility
              'params': {
              'targetPage': 'funA' // 在EntryAbility中处理这个信息
            }
        });
    })
 }
...
```



卡片事件 - router (2)

- 在UIAbility的EntryAbility.ts文件中接收router事件并获取参数，根据传递的params不同，选择拉起不同的页面。

```
let selectPage = ""; let currentWindowStage = null;
export default class CameraAbility extends UIAbility {
  // 如果UIAbility第一次启动，在收到Router事件后会触发onCreate生命周期回调
  onCreate(want, launchParam) {
    // 获取router事件中传递的targetPage参数
if (want.parameters.params !== undefined) {
    selectPage = JSON.parse(want.parameters.params).targetPage;
    }
  }
  // 如果UIAbility已在后台运行，在收到Router事件后会触发onNewWant生命周期回调
  onNewWant(want, launchParam) {
      ...
      if (currentWindowStage != null) {
      this.onWindowStageCreate(currentWindowStage);
    }
  }
};
```

```
 onWindowStageCreate(windowStage:
window.WindowStage) {
    let targetPage;
   // 根据传递的targetPage不同，选择拉起不同的页面
   switch (selectPage) {
     case 'funA':
       targetPage = 'pages/FunA';
       break;
       ...
}
    if (currentWindowStage === null) {
      currentWindowStage = windowStage;
    }
    windowStage.loadContent(targetPage, (err, data)
=> {
        ...
    });
  }
```

卡片事件 - call (1)

- 使用call事件拉起指定UIAbility到后台。
- 许多应用希望借助卡片的能力，实现和应用在前台时相同的功能。例如音乐卡片，卡片上提供播放、暂停等按钮，点击不同按钮将触发音乐应用的不同功能。在卡片页面发送call事件示例如下：

```
//卡片页面文件WidegetCard.ets
 ...
struct WidgetCard {
  build() {
      Column() {
          Button('功能A')
              .onClick(() => {
                postCardAction(this, {
                      'action': 'call',
                      'abilityName': 'EntryAbility',
                      'params': {
                      'method': 'funA' // 在EntryAbility中调用的方法名
                    }
              });
          })
    }
  }
}
```

卡片事件 - call (2)

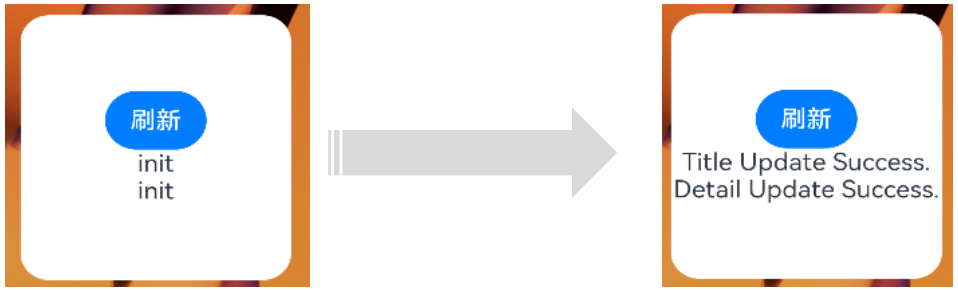
- 在UIAbility的EntryAbility.ts文件中接收call事件并获取参数，根据传递的method不同，执行不同的方法；其余数据可以通过readString的方式获取。需要注意的是，UIAbility需要onCreate生命周期中监听所需的方法。

```
function FunACall(data) { // 获取call事件中传递的所有参数
  ...
}
export default class CameraAbility extends UIAbility { // 如果UIAbility第一次启动，在收到call事件后会触发onCreate生命周期回调
onCreate(want, launchParam) {
    try { // 监听call事件所需的方法
            this.callee.on('funA', FunACall);
     } catch (error) {
          ...
     }
 }
 onDestroy() { // 进程退出时，解除监听
    try {
       this.callee.off('funA');
   } catch (error) {
        ...
     }
  }
};
```

卡片事件 - message刷新卡片（1）

- 通过message事件刷新卡片内容。
  - message、router、call均能刷新卡片内容，且核心代码是一致的。

```
let storage = new LocalStorage();
@Entry(storage)
@Component
struct WidgetCard {
  @LocalStorageProp('title') title: string = 'init';
  @LocalStorageProp('detail') detail: string = 'init';
  build() {
    Column() {
      Button('刷新')
        .onClick(() => {
          postCardAction(this, {
            'action':'message',
            'params': {'msgTest':'messageEvent’}
          });
        })
      Text(`${this.title}`)
      Text(`${this.detail}`)
    }
```



HUAWEI

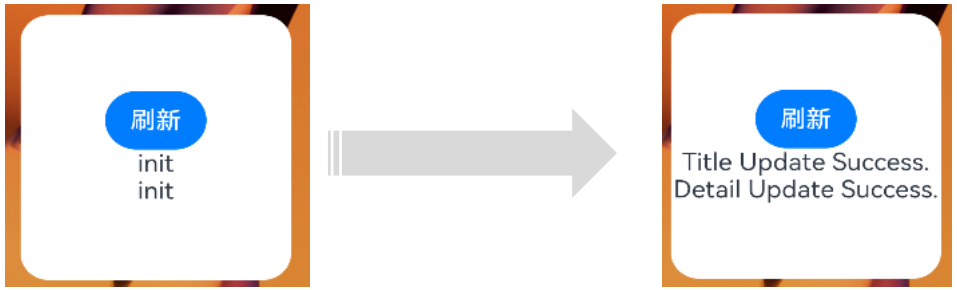
卡片事件 - message刷新卡片 (2)

- 在EntryFormAbility.ts文件中的onFormEvent生命周期中调用updateForm接口刷新卡片。

```
import formBindingData from '@ohos.app.form.formBindingData';
import FormExtensionAbility from '@ohos.app.form.FormExtensionAbility';
import formProvider from '@ohos.app.form.formProvider';

export default class EntryFormAbility extends FormExtensionAbility {
  onFormEvent(formId, message) {
   let formData = {
      'title': 'Title Update Success.', // 和卡片布局中对应
      'detail' : 'Detail Update Success.', }; // 和卡片布局中对应

    let formInfo = formBindingData.createFormBindingData(formData)
    formProvider.updateForm(formId, formInfo).then((data) => {
      ...
    }).catch((error) => {
      ...
    })
  }
  ...
}
```



卡片事件 - 数据交互 (1)

- 有时在刷新卡片内容时，我们希望实现定时刷新或者定点刷新，无需手动刷新。
  - 定时刷新：表示在一定时间间隔内调用onUpdateForm的生命周期回调函数自动刷新卡片内容。可以在form_config.json配置文件的updateDuration字段中进行设置。
  - 定点刷新：表示每天在某个时间点刷新，在form_config.json文件的scheduledUpdateTime字段进行设置。

```
{
  "forms": [
    {
      ...
      "updateEnabled ": true, // 开启刷新功能
      "scheduledUpdateTime": "10:30",
      "updateDuration": 2, // 设置卡片定时刷新的更新周期（单位为30分钟，取值为自然数）
      "defaultDimension": "2*2",
      "supportDimensions": ["2*2" ]
    }
  ]
}
```

1 同时配置定时刷新和定点刷新时，定时刷新优先级更高

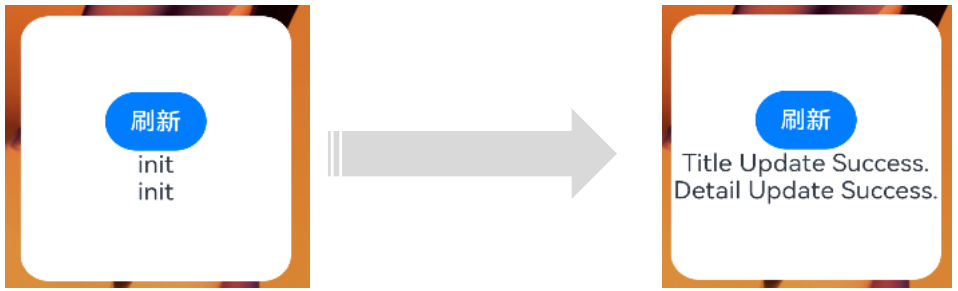
2 如配置定点刷新，则需将updateDuration配置为0

卡片事件 - 数据交互 (2)

- 在触发定时或定点刷新后，系统会调用EntryFormAbility.ts文件中的生命周期回调onUpdateForm ，在回调中可以使用updateForm进行提供方刷新卡片。

```
...
export default class EntryFormAbility extends FormExtensionAbility {
  onUpdateForm(formId) {
   let formData = {
      'title': 'Title Update Success.', // 和卡片布局中对应
      'detail' : 'Detail Update Success.',
       };
    let formInfo = formBindingData.createFormBindingData(formData)
    formProvider.updateForm(formId, formInfo).then((data) => {
      ...
    }).catch((error) => {
      ...
    })
  }
  ...
}
```

此时无需点击刷新按钮，会自动定点或定时刷新



本章小结

- 本章主要介绍了服务卡片的相关概念和基于ArkTS UI的卡片开发流程 。
- 讲解了服务卡片的概述：定义、优势、使用场景等。
- 讲解了ArkTS卡片的运行机制：实现原理与渲染服务原理。
- 讲解了ArkTS卡片的开发流程：创建过程、卡片配置、生命周期、事件、以及刷新机制等。

思考题

1. （单选题）若卡片支持定时更新功能，则提供方需要重写哪个方法以支持数据更新？（ ）

   A. onChangeFormVisibility(newStatus)

   B. onUpdateForm(formId)

   C. onAddForm(want)

   D. onCastToNormalForm(formId)

学习推荐

- 官方学习网站
  - HarmonyOS官网：https://developer.harmonyos.com/
  - HarmonyOS应用开发文档： https://developer.huawei.com/consumer/cn/
  - OpenHarmony官网：https://edu.huaweicloud.com/
  - 华为开发者论坛： https://developer.huawei.com/consumer/cn/forum/

华为云开发者学堂

感谢