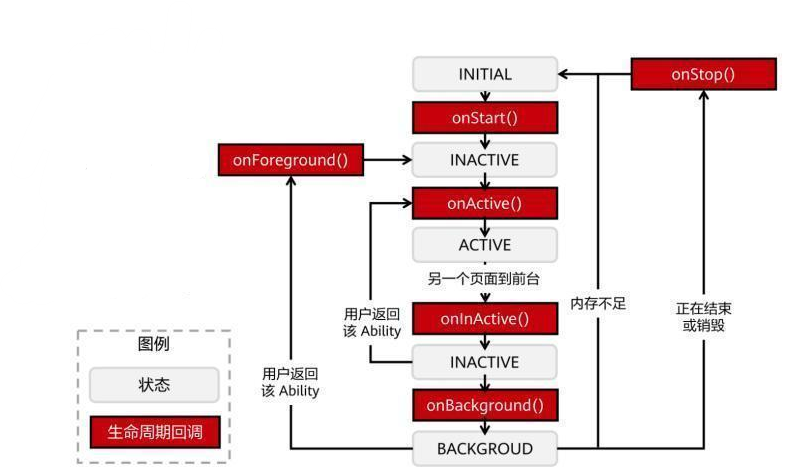
# Feature Ability（有UI界面）

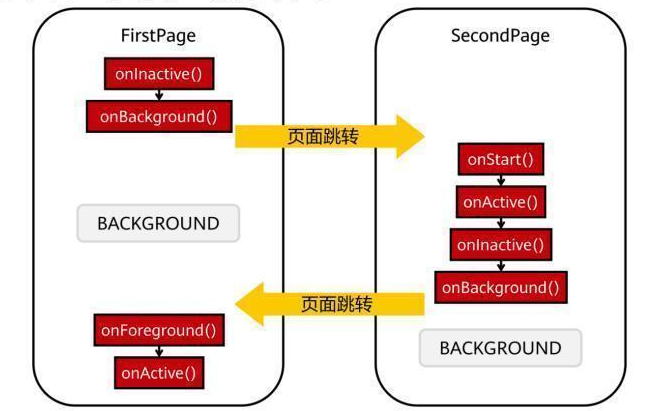
* Page模板（Page Ability，以下简称“Page”）是Feature Ability唯一支持的模板，用于提供与用户交互的能力
* 元程序：拥有UI的单一特性程序实体
* 一个Page可以由一个或多个AbilitySlice构成，AbilitySlice是指应用的单个页面及其控制逻辑的总和
  + Page：UI界面
  + Slice：子页面
* 当一个Page由多个AbilitySlice共同构成时，这些AbilitySlice页面提供的业务能力应具有高度相关性

## Page Ability生命周期

* 当用户进入、浏览、退出页面时，页面将有不同的状态，并回调不同的生命周期状态方法给外界
* Page Ability 生命回调方法（AbilitySlice与之相似，也可参考下图）：



* onStart()
  + 当系统首次创建Page实例时，触发该回调
  + 对于一个Page实例，该回调在其生命周期过程中仅触发一次，Page在该逻辑后将进入INACTIVE状态，开发者必须重写该方法，并在此配置默认展示的AbilitySlice
* onActive()
  + Page会在进入INACTIVE状态后来到前台，然后系统调用此回调
  + Page在此之后进入ACTIVE状态，该状态是应用与用户交互的状态。
  + Page将保持在此状态，除非某类事件发生导致Page失去焦点，比如用户点击返回键或导航到其他Page。当此类事件发生时，会触发Page回到INACTIVE状态，系统将调用onInactive()回调。此后，Page可能重新回到ACTIVE状态，系统将再次调用onActive()回调。因此，开发者通常需要成对实现onActive()和onInactive()，并在onActive()中获取在onInactive()中被释放的资源
* onInactive()
  + 当Page失去焦点时，系统将调用此回调，此后Page进入INACTIVE状态
  + 开发者可以在此回调中实现Page失去焦点时应表现的恰当行为
* onBackground()
  + Page不再对用户可见，系统将调用此回调通知开发者用户进行相应的资源释放，此后Page进入BACKGROUND状态
  + 开发者应该在此回调中释放Page不可见时无用的资源，或在此回调中执行较为耗时的状态保存操作
* onForeground()
  + 处于BACKGROUND状态的Page仍然驻留在内存中，当重新回到前台时（比如用户重新导航到此Page），系统将先调用onForeground()回调通知开发者。而后Page的生命周期状态回到INACTIVE状态
  + 开发者应当在此回调中重新申请在onBackground()中释放的资源，最后Page的生命周期状态进一步回到ACTIVE状态，系统将通过onActive()回调通知开发者用户
* onStop()
  + 系统将要销毁Page时，将会触发此回调函数，通知用户进行系统资源的释放
  + 销毁Page的可能原因包括以下几个方面：
    - 用户通过系统管理能力关闭指定Page，例如使用任务管理器关闭Page
    - 用户行为触发Page的terminateAbility()方法调用，例如使用应用的退出功能
    - 配置变更导致系统暂时销毁Page并重建
    - 系统出于资源管理目的，自动触发对处于BACKGROUND状态Page的销毁
* 系统管理或用户操作等行为均会引起Page实例在其生命周期的不同状态之间进行转换。Ability类提供的回调机制能够让Page及时感知外界变化，从而正确地应对状态变化（比如释放资源），这有助于提升应用的性能和稳健性
* 不同Page页面互相跳转时的生命周期状态变化如图所示



## Page与AbilitySllice生命周期关联：

* 当AbilitySlice处于前台且具有焦点时，其生命周期状态随着所属Page的生命周期状态的变化而变化
* 当一个Page拥有多个AbilitySlice时，例如：MyAbility下有FooAbilitySlice和BarAbilitySlice，当前FooAbilitySlice处于前台并获得焦点，并即将导航到BarAbilitySlice，在此期间的生命周期状态变化顺序为：

1. FooAbilitySlice从ACTIVE状态变为INACTIVE状态
2. BarAbilitySlice则从INITIAL状态首先变为INACTIVE状态，然后变为ACTIVE状态（假定此前BarAbilitySlice未曾启动）
3. FooAbilitySlice从INACTIVE状态变为BACKGROUND状态

## AbilitySlice生命周期

* AbilitySlice 作为 Page 的组成单元，其生命周期是依托于其所属 Page 生命周期的
* AbilitySlice 和 Page 具有相同的生命周期状态和同名的回调，当 Page 生命周期发生变化时，它的 AbilitySlice 也会发生相同的生命周期变化
* AbilitySlice 具有独立于 Page 的生命周期变化，这发生在同一 Page 中的 AbilitySlice 之间导航时，此时 Page 的生命周期状态不会改变
* 由于 AbilitySlice 承载具体的页面，开发者必须重写 AbilitySlice 的 onStart()回调，并在此方法中通过setUIContent()方法设置页面AbilitySlice 实例创建和管理通常由应用负责，系统仅在特定情况下会创建 AbilitySlice 实例。例如，通过导航启动某个 AbilitySlice 时，是由系统负责实例化；但是在同一个 Page 中不同的 AbilitySlice 间导航时则由应用负责实例化

## AbilitySlice路由设置

* 配置页面路由：虽然一个Page可以包含多个AbilitySlice，但是Page进入前台时界面默认只展示一个AbilitySlice
  + 默认展示的AbilitySlice是通过setMainRoute()方法来指定的
  + 如果需要更改默认展示的AbilitySlice，可以通过addActionRoute()方法为此AbilitySlice配置一条路由规则
* addActionRoute()方法中使用的动作命名，需要在应用配置文件（config.json）中注册
* 当其他Page实例期望导航到此AbilitySlice时，可以在Intent中指定Action

## AbilitySlice间导航

* AbilitySlice同一Page内导航：
  + 当发起导航的AbilitySlice和导航目标的AbilitySlice处于同一个Page时，您可以通过present()方法实现导航
  + 如果开发者希望在用户从导航目标AbilitySlice返回时，能够获得其返回结果，则应当使用presentForResult()实现导航
  + 用户从导航目标AbilitySlice返回时，系统将回调onResult()来接收和处理返回结果，开发者需要重写该方法
  + 返回结果由导航目标AbilitySlice在其生命周期内通过setResult()进行设置
  + 系统为每个Page维护了一个AbilitySlice实例的栈，每个进入前台的AbilitySlice实例均会入栈
  + 当开发者在调用present()或presentForResult()时指定的AbilitySlice实例已经在栈中存在时，则栈中位于此实例之上的AbilitySlice均会出栈并终止其生命周期

## AbilitySlice不同Page间导航

* 不同Page之间可以跳转，并可以指定跳转到目标Page中某个具体的AbilitySlice
* AbilitySlice作为Page的内部单元，以Action的形式对外暴露，因此可以通过配置Intent的Action导航到目标AbilitySlice
* Page间的导航可以使用startAbility()或startAbilityForResult()方法，获得返回结果的回调为onAbilityResult()
* 在Ability中调用setResult()可以设置返回结果
* AbilitySlice 作为 Page 的内部单元，以 Action 的形式对外暴露，因此可以通过配置Intent 的 Action 导航到目标 AbilitySlice
* Page 间的导航可以使用 startAbility()或startAbilityForResult()方法，获得返回结果的回调为 onAbilityResult()。在 Ability 中调用
* setResult()可以设置返回结果

## 跨设备迁移

* 跨设备迁移（下文简称“迁移”）支持将Page在同一用户的不同设备间迁移，以便支持用户无缝切换的诉求
* 以Page从设备A迁移到设备B为例，迁移动作主要步骤如下：

1. 设备A上的Page请求迁移
2. HarmonyOS处理迁移任务，并回调设备A上Page的保存数据方法，用于保存迁移必须的数据
3. HarmonyOS在设备B上启动同一个Page，并回调其恢复数据方法

* 一个应用可能包含多个Page，仅需要在支持迁移的Page中通过以下方法实现IAbilityContinuation接口。同时，此Page所包含的所有AbilitySlice也需要实现此接口
* 实现IAbilityContinuation接口：
  + onStartContinuation()：Page请求迁移后，系统首先回调此方法，开发者可以在此回调中决策当前是否可以执行迁移，比如，弹框让用户确认是否开始迁移
  + onSaveData()：如果onStartContinuation()返回true，则系统回调此方法，开发者在此回调中保存必须传递到另外设备上以便恢复Page状态的数据
  + onRestoreData()：源侧设备上Page完成保存数据后，系统在目标侧设备上回调此方法，开发者在此回调中接受用于恢复Page状态的数据。注意，在目标侧设备上的Page会重新启动其生命周期，无论其启动模式如何配置。且系统回调此方法的时机在onStart()之前
  + onCompleteContinuation()：目标侧设备上恢复数据一旦完成，系统就会在源侧设备上回调Page的此方法，以便通知应用迁移流程已结束。开发者可以在此检查迁移结果是否成功，并在此处理迁移结束的动作，例如，应用可以在迁移完成后终止自身生命周期
  + onFailedContinuation()：迁移过程中发生异常，系统会在发起端设备上回调FA的此方法，以便通知应用迁移流程发生的异常。并不是所有异常都会回调FA此方法，仅局限于该接口枚举的异常。开发者可以在此检查异常信息，并在此处理迁移异常发生后的动作，例如，应用可以提醒用户此时发生的异常信息
  + onRemoteTerminated()：如果开发者使用continueAbilityReversibly()而不是continueAbility()，则此后可以在源侧设备上使用reverseContinueAbility()进行回迁。这种场景下，相当于同一个Page（的两个实例）同时在两个设备上运行，迁移完成后，如果目标侧设备上Page因任何原因终止，则源侧Page通过此回调接收终止通知

## 请求迁移

* 实现IAbilityContinuation的Page可以在其生命周期内，调用continueAbility()或continueAbilityReversibly()请求迁移。两者的区别是，通过continueAbilityReversibly()发起的迁移此后可以进行回迁
* 以Page从设备A迁移到设备B为例，详细的流程如下：

1. 设备A上的Page请求迁移
2. 系统回调设备A上Page及其AbilitySlice栈中所有AbilitySlice实例的AbilityContinuation.onStartContinuation()方法，以确认当前是否可以立即迁移（即做迁移前的准备工作）
3. 如果可以立即迁移，则系统回调设备A上Page及其AbilitySlice栈中所有AbilitySlice实例的IAbilityContinuation.onSaveData()方法，以便保存迁移后恢复状态必须的数据（即保存迁移所需的数据）
4. 如果保存数据成功，则系统在设备B上启动同一个Page，并恢复AbilitySlice栈，然后回调IAbilityContinuation.onRestoreData()方法，传递此前保存的数据；此后设备B上此Page从onStart()开始其生命周期回调（即恢复迁移后所需要的数据）
5. 系统回调设备A上Page及其AbilitySlice栈中所有AbilitySlice实例的IAbilityContinuation.onCompleteContinuation()方法，通知数据恢复成功与否（即用于完成迁移后的工作）
6. 迁移过程中发生异常，系统回调设备A上Page及其AbilitySlice栈中所有AbilitySlice实例的IAbilityContinuation.onFailedContinuation()方法，通知迁移过程中发生异常，并不是所有异常都会回调FA此方法，仅局限于该接口枚举的异常

* 迁移相关方法：
  + 启动迁移：continueAbility()
  + 回迁：reverseContinueAbility()
  + 迁移过程事件回调：
* onStartContinuation()
* onSaveData()
* onRestoreData()
* onCompleteContinuation()
* onRemoteTerminated()

## 请求回签

* 使用continueAbilityReversibly()请求迁移并完成后，源侧设备上已迁移的Page可以发起回迁，以便使用户活动重新回到此设备
* 以Page从设备A迁移到设备B后并请求回迁为例，详细的流程如下：
  + 设备A上的Page请求回迁
  + 系统回调设备B上Page及其AbilitySlice栈中所有AbilitySlice实例的IAbilityContinuation.onStartContinuation()方法，以确认当前是否可以立即迁移
  + 如果可以立即迁移，则系统回调设备B上Page及其AbilitySlice栈中所有AbilitySlice实例的IAbilityContinuation.onSaveData()方法，以便保存回迁后恢复状态必须的数据
  + 如果保存数据成功，则系统在设备A上Page恢复AbilitySlice栈，然后回调IAbilityContinuation.onRestoreData()方法，传递此前保存的数据
  + 如果数据恢复成功，则系统终止设备B上Page的生命周期