

```

-----
-- callog.lua
-- by Stewart Bracken http://stewart.bracken.bz stew.bracken@gmail.com
-- List call logs from Twilio with Corona widgets.
-- Lots of code leveraged from WidgetDemo.
-- Known bug: if you switch to another tab before if finished retriving calls,
-- this get funky.
-- Known critical bug: text labels and buttons are recreated on every scene
-- enter but never destroyed, adding up a lot of widgets and whatnot.
-----

local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

-- create a constant for the left spacing of the row content
local LEFT_PADDING = 10

local tableView = nil

local isExiting = false

function scene:createTableLog(twilio_response)
    -- Forward reference for our tableView
    local group = self.view

    --Data to be displayed when a row is selected
    local rowDisplayData = {['To:'] = 'to_formatted', ['From:'] = 'from_formatted',
    ['Status:'] = 'status', ['Duration:'] = 'duration'}

    -- Text to show which item we selected
    local itemSelected = {}
    local itemY = 150
    for k, v in pairs(rowDisplayData) do
        itemSelected[k] = display.newText( k, 0, 0, native.systemFontBold, 14 )
        itemSelected[k]:setTextColor( 0 )
        itemSelected[k].x = display.contentWidth + itemSelected[k].contentWidth * 0.5
        itemSelected[k].y = itemY
        group:insert( itemSelected[k] )
        itemY = itemY + itemSelected[k].contentHeight + 10
    end
    local selectedItemAnchor = itemSelected['To:']

    -- Function to return to the list
    local function goBack( event )

```

```

--Transition in the list, transition out the item selected text and the
back button
    transition.to( tableView, { x = 0, time = 400, transition = easing.outExpo
} )

    for k, v in pairs(itemSelected) do
        local item = v
        transition.to( item, { x = display.contentWidth + item.contentWidth *
0.5, time = 400, transition = easing.outExpo } )

    end
    transition.to( event.target, { x = display.contentWidth +
event.target.contentWidth * 0.5, time = 400, transition = easing.outQuad } )
end

-- Back button
local backButton = widget.newButton
{
    width = 198,
    height = 59,
    label = "Back",
    onRelease = goBack,
}
backButton.x = display.contentWidth + backButton.contentWidth * 0.5
backButton.y = selectedItemAnchor.y + selectedItemAnchor.contentHeight +
backButton.contentHeight
group:insert( backButton )

-- Handle row rendering
local function onRowRender( event )
    local phase = event.phase
    local row = event.row
    local text = "Call log page 1. All pages not shown in ex."
    if not row.isCategory then
        text = row.params.sid
    end

    local rowTitle = display.newText( row, text, 0, 0, nil, 14 )
    rowTitle.x = row.x - ( row.contentWidth * 0.5 ) + ( rowTitle.contentWidth *
0.5 ) + LEFT_PADDING
    rowTitle.y = row.contentHeight * 0.5
    rowTitle:setTextColor( 0, 0, 0 )

end

```

```

-- Handle touches on the row
local function onRowTouch( event )
    local phase = event.phase
    local row = event.target

    if "release" == phase then
        transition.to( tableView, { x = - tableView.contentWidth, time = 400,
transition = easing.outExpo } )
        transition.to( backButton, { x = display.contentCenterX, time = 400,
transition = easing.outQuad } )
        for k, v in pairs(itemSelected) do
            local item = v
            --Update the item selected text
            item.text = k.." "..row.params[rowDisplayData[k]]

            --Transition out the list, transition in the item selected text and
the back button

            transition.to( item, { x = display.contentCenterX, time = 400,
transition = easing.outExpo } )

        end
    end
end

-- Create a tableView
tableView = widget.newTableView
{
    top = 32,
    width = 320,
    height = 400,
    --listener = tableViewListener,
    onRowRender = onRowRender,
    --onRowUpdate = onRowUpdate,
    onRowTouch = onRowTouch,
}
group:insert( tableView )

-- Insert the row into the tableView
tableView:insertRow
{
    isCategory = true,
    rowHeight = 35,

```

```

rowColor = {default = { 150, 160, 180, 200 }, over = { 30, 144, 255 }, },
lineColor = { 220, 220, 220 }
}

-- Create row per call log
for i, call in ipairs(twilio_response.calls) do
    local rowColor =
    {
        default = { 255, 255, 255 },
        over = { 30, 144, 255 },
    }

    -- Insert the row into the tableView
    tableView:insertRow
    {
        isCategory = false,
        rowHeight = 40,
        rowColor = rowColor,
        lineColor = { 220, 220, 220 },
        params = call
    }
end

function scene:enterScene(event)
    isExiting = false
    tableView = nil
    local group = self.view

    -- Create a spinner widget
    local spinner = widget.newSpinner
    {
        left = 150,
        top = 200,
    }
    group:insert( spinner )

    -- Start the spinner animating
    spinner:start()
    spinner.isVisible = true

    local statusText = display.newText( "Retriving Twilio call logs", 60, 240,
native.systemFont, 20 )
    --statusText.x = 10

```

```
--statusText.y = 235
statusText:setTextColor(0, 0, 0)
group:insert( statusText )

local function request_listener(event)
    if event.success and not isExiting then
        group:remove(spinner)
        spinner:removeSelf()
        spinner=nil
        statusText:removeSelf()
        group:remove(statusText)
        self:createTableLog(event.response)
    else
        statusText.text = "Error retriving Twilio call logs"
    end
end

local vars = {Type="Calls"} --retrieve all call logs
R:request(vars, "GET", request_listener)
end

-- Our scene
function scene:exitScene( event )
    isExiting=true
    if tableView then
        tableView:removeSelf()
        self.view:remove(tableView)
        tableView = nil
    end
end

scene:addEventListener("enterScene")
scene:addEventListener("exitScene")

return scene
```