Walkthrough - Saving the Activity state

Walkthrough

Open the project *ActivityLifecycle_Start*, and run it once. This is a very simple project that has two activities and will demonstrate the activity lifecycle and how the various lifecycle methods are called.

Let’s change the application so that we have a button that will count the number of times it was clicked. We will save the number of click counts to instance state so that they are not lost if the application is destroyed.

Lets start making changes to our application.

1. Add an instance variable to `MainActivity`:

   ```
   int _counter = 0;
   ```

2. Next, override the method as show in the following code snippet:

   ```
   protected override void OnSaveInstanceState (Bundle outState)
   {
       outState.PutInt ("click_count", _counter);
       Log.Debug(GetType().FullName, "Saving instance state");

       base.OnSaveInstanceState (outState);
   }
   ```

   This code will save the value of the variable `_counter` to the bundle.

3. Next, edit the layout file `Resource/layout/Main.axml` to resemble the following XML:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   ```

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
        android:id="@+id/myButton"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/mybutton_text" />
    <Button
        android:id="@+id/clickButton"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/counterbutton_text" />
</LinearLayout>
```

This adds a new button to the layout – one that will display the number of times the user has clicked the button.

4. The next change we need to make to our code is more significant. Change the `OnCreate` method to resemble the following code:

```
protected override void OnCreate(Bundle bundle)
{
    Log.Debug(GetType().FullName, "Activity A - OnCreate");
    base.OnCreate(bundle);
    SetContentView (Resource.Layout.Main);
    FindViewById<Button>(Resource.Id.myButton).Click += (sender,
args) => {
        var intent = new Intent(this, typeof(SecondActivity));
        StartActivity(intent);
    };
    if (bundle != null)
    {
        _counter = bundle.GetInt ("click_count", 0);
        Log.Debug(GetType().FullName, "Recovered instance
state");
    }

    var clickbutton = FindViewById<Button>
(Resource.Id.clickButton);
    clickbutton.Text =
Resources.GetString(Resource.String.counterbutton_text,
_counter);
    clickbutton.Click += (object sender, System.EventArgs e) =>
{
        _counter++;
        clickbutton.Text =
Resources.GetString(Resource.String.counterbutton_text,
_counter);
    };
}
```

There is a lot happening in this code, so lets take a minute to explain what his happening. First we check to see if the bundle parameter is null. If it isn't, then we try to extract the value of the key `click_count` from it.

Next we wire up some functionality to the `clickButton` in the layout – we will update our `_counter` parameter each time the button is clicked by the user.

5. With this code in place, the activity now needs to update its instance state. Add the following method to `MainActivity`:

```
protected override void OnSaveInstanceState (Bundle outState)
{
    outState.PutInt ("click_count", _counter);
```

```
        Log.Debug(GetType().FullName, "Saving instance state");
        base.OnSaveInstanceState (outState);
    }
```

This code will save the value of the variable `_counter` to the instance state.

6. Run the application, and pay attention to the application output as it is running. You should see the `Log.Debug` messages appearing as the application is running.

   At this point you should have a fundamental understanding of the Activity Lifecycle and the callback methods.

Summary

In this walkthough, we have used our knowledge of the Activity Lifecycle to preserve state data.