Runtime Configuration Changes

Android supports runtime changes to the language, location, and hardware by terminating and restarting each application and reloading the resource values.

This default behavior isn't always convenient or desirable, particularly as some confi guration changes (like screen orientation and keyboard visibility) can occur as easily as a user rotating the device or sliding out the keyboard. You can customize your application's response to these changes by detecting and reacting to them yourself.

To have an Activity listen for runtime confi guration changes, add an android:configChanges attribute to its manifest node, specifying the confi guration changes you want to handle.

The following list describes the confi guration changes you can specify:
❑ orientation The screen has been rotated between portrait and landscape.
❑ keyboardHidden The keyboard has been exposed or hidden.
❑ fontScale The user has changed the preferred font size.
❑ locale The user has chosen a different language setting.
❑ keyboard The type of keyboard has changed; for example, the phone may have a 12 keypad that fl ips out to reveal a full keyboard.
❑ touchscreen or navigation The type of keyboard or navigation method has changed. Neither of these events should normally happen.

You can select multiple events to handle by separating the values with a pipe (|).
The following XML snippet shows an activity node declaring that it will handle changes in screen orientation and keyboard visibility:

```
<activity android:name=".TodoList"
android:label="@string/app_name"
android:theme="@style/TodoTheme"
android:configChanges="orientation|keyboard"/>
```

Adding this attribute suppresses the restart for the specifi ed confi guration changes, instead, triggering the on Configuration Changed method in the Activity. Override this method to handle the confi guration changes using the passed-in Configuration object to determine the new confi guration values, as shown in the following skeleton code. Be sure to call back to the super class and reload any resource values that the Activity uses in case they've changed.

```
@Override
public void onConfigurationChanged(Configuration _newConfig) {
super.onConfigurationChanged(_newConfig);
[ ... Update any UI based on resource values ... ]
if (_newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
[ ... React to different orientation ... ]
}
if (_newConfig.keyboardHidden == Configuration.KEYBOARDHIDDEN_NO) {
[ ... React to changed keyboard visibility ... ]
}
}
```

When onConfigurationChanged is called, the Activity's Resource variables will have already been updated with the new values so they'll be safe to use.

Any confi guration change that you don't explicitly fl ag as being handled by your application will still cause an application restart without a call to onConfigurationChanged.