Archived:Dynamic Layout handling with QWidget

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (based on QWidget) is deprecated.

Dynamic Layout handling with QWidget

To be able to catch the layout change event, you first need to install event filter, which will give you then the events when layout changes happen. This can be done for example in the main.cpp, in where you can use installEventFilter function of the QApplication to install a filter, for example like this:

```
QApplication a(argc, argv);
QtEggAlarm w;
a.installEventFilter(&w);
```

The event filter class then needs to implement eventFilter function which will then be called when events happen. For layout changes a QEvent::Resize event will be generated, thus you could catch the event in the event filter as follows:

```
bool QtEggAlarm::eventFilter(QObject* /*receiver*/, QEvent* event)
{
 if(event->type() == QEvent::Resize)
 {
  SetScreenSizeAndPosition();
  return true;
 }


  return false;
}
```

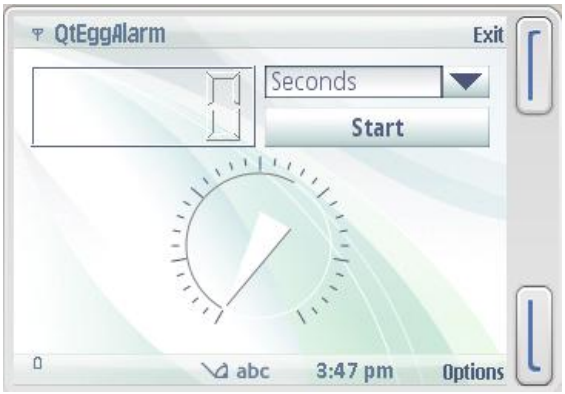
Note that if the return value is false, then the event will be given to other filters, and if it is true, it will not be forwarded to other components.

Then If you want to Widget to fill the whole client area in S60 device, the SetScreenSizeAndPosition function could be implemented as follows:

```
void QtEggAlarm::SetScreenSizeAndPosition(void)
{
 setGeometry(QApplication::desktop()->availableGeometry());
}
```

Screen shots

(C) Copyright Nokia 2014. All rights reserved.



Example project

File:QtEggAlarm.zip

(C) Copyright Nokia 2014. All rights reserved.