

Re: Generic barcode scanner programming

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.compactframework/2007-02/msg001>

- *From:* "Neil Cowburn" <[neilc@\[nospamthanks\].opennetcf.com](mailto:neilc@[nospamthanks].opennetcf.com)>
 - *Date:* Wed, 7 Feb 2007 15:12:01 -0000
-

In this scenario, design patterns are king. In particular, factory patterns. By defining a common interface, you can create simple wrappers for each of the scanner SDKs. Then you create a factory method that simply returns a reference to the common interface. In your app, you call the factory method to get a reference to the relevant scanner wrapper and then just code against the interface. Some call this a provider model. Some call it a plugin model. Which you prefer, it does the job of abstracting the scanner SDK implementation away from the application allowing your app to use pretty much SDK you like as long as you provide a wrapper for the SDK which implements your common interface.

--
Neil Cowburn
Principal Partner
OpenNETCF Consulting, LLC.

<http://www.opennetcf.com/>

"Matt Lacey" <m.lacey@xxxxxxxxxxxxx> wrote in message
<news:1170852895.084696.134840@xx>

On 7 Feb, 12:38, "ac0203" <achap...@xxxxxxxxxxxxx> wrote:

We currently have a Symbol device (MC1000) and have used VS2005 to write an application which captures data through the built-in scanner. I have used the SDK from Symbol to get the controls in .NET and this is fine.

This is fine so long as we stay with Symbol devices. What happens if we install onto a different device (not Symbol). I presume that the Symbol libraries will not work on other devices and we would have to change our app to use the library from the new device.

It would be nice to have some generic code that we could use so that our app can be put onto any device running CE. Has anyone done this? This would save me a whole load of hassle as it looks like our customer may have their own devices in some cases and would want our app to run on them rather than go and buy new devices.

Cheers

AC

The bad news is that you will need different code for each type of scanner you are using.

This is because different scanners/devices will require different supporting files (dlls, normally). They will also implement the way of returning what was scanned (normally barcode contents, but may be an image). Configuration of the scanners can also vary widely.

In the past I have developed an application that included scanning and imaging functionality that ran on devices from Symbol, Handheld, Intermec, Psion Teklogix and NordicID. We did this by putting all the device specific code in separate classes and then used compiler directives to include the relevant classes at build time. This did, obviously, require that we had separate exes for the different machines.

Most device manufacturers have good developer tools and are keen to help as any programs you write that will work on their devices can only lead to the sale of more of their devices.

I know of other people who have created an abstraction layer and then determined in code which device the code was running on and used the relevant classes accordingly.

Yes it would be lovely if all scanners worked the same way but this would cause at least two problems (more may come to me if I think about it longer.) Firstly, device manufacturers would lose their tie ins with their customers. Secondly, different scanners work in different ways and can return different information. A standard interface to ALL possible events, parameters and return data would be VERY complicated.