Automatically edit project configuration in Nokia IDE at build-time

This article shows how to create and run a (C#) program at build time to control your Nokia IDE project configuration.

Introduction

The Nokia IDE is based on Eclipse, which among other things means that we can create and set programs to be run as part of the build process. This gives us the opportunity to automatically alter configuration files that we couldn't touch through our code.

This article shows how we can use this feature to automatically set an appropriate icon in our JAD file for the platform guidelines of the current target. We will first create the program that does this job and then set it to handle required changes right when our project is building.

The program

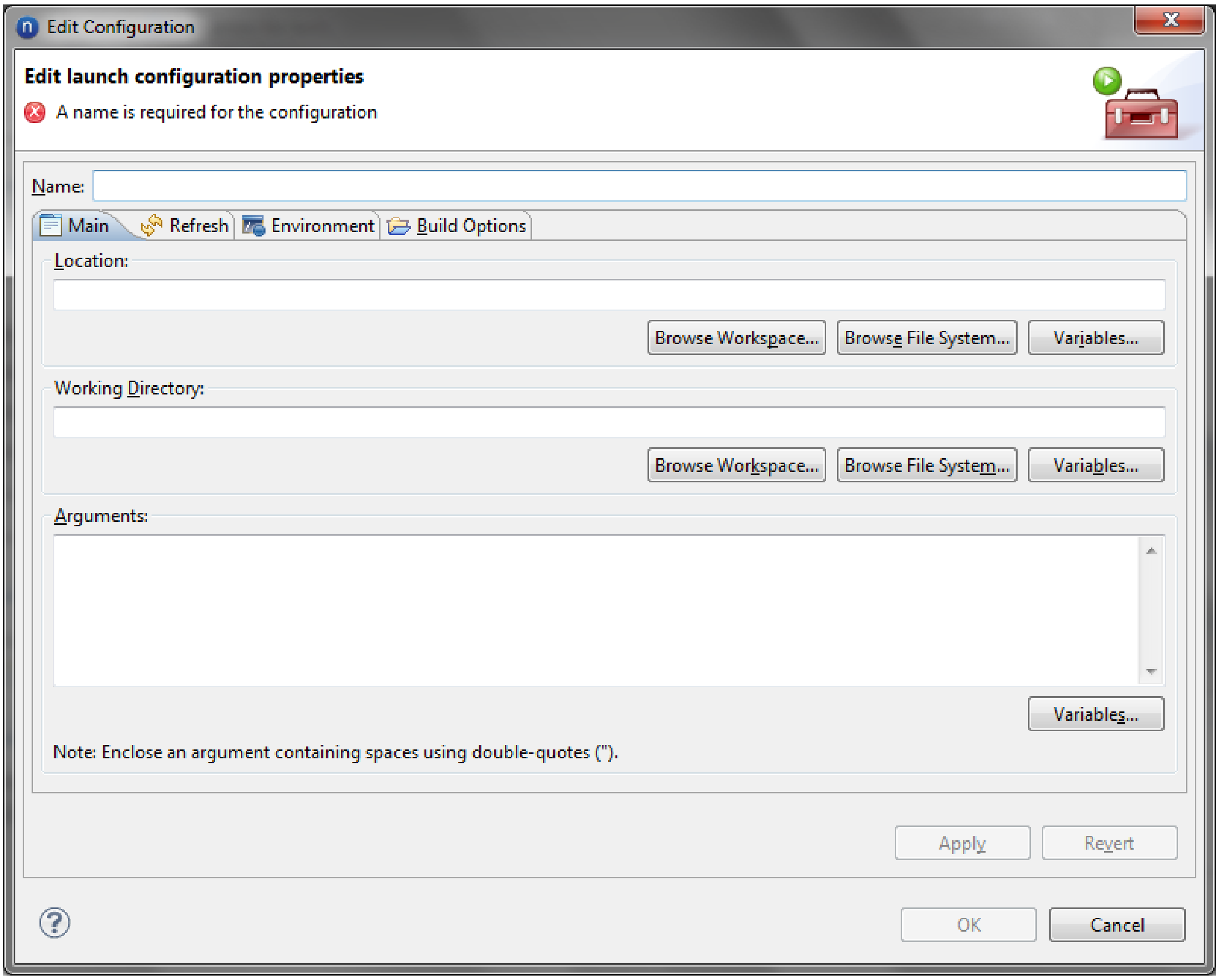
Then configuration file that contains the descriptions of our projects targeting platforms is the *.mtj*. This also contains the platform that we are currently targeting marked as *active*.

We use *LINQ* to query this XML file and find the active configuration. Then we store the platform’s name in our `platform` string variable. After we get the name we call the `SetIcon` function to make the desired changed in our application’s .jad file.

`SetIcon` searches for the line in the JAD file that contains the information about the MIDlets icon and changes it to that of the passed string `iconPath` argument.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Xml.Linq;

namespace NokiaPlatformScript
{
    class Program
    {
        static void Main(string[] args)
        {
            XElement root = XElement.Load(".mtj");
            IEnumerable<XElement> activeConfigs =
                from config in root.Descendants("configuration")
                where (string)config.Attribute("active") == "true"
                select config;

            if (activeConfigs.Count() == 1)
            {
                string
platform=((XElement)activeConfigs.ElementAt(0)).Attribute("name").Value;

                switch (platform)
                {
                    case "Nokia SDK 1.0 for Java": //Touch&Type
                        SetIcon("/images/logo1.png");
                        break;
                    case "Nokia SDK 2.0 for Java": //Full Touch
                        SetIcon("/images/logo2.png");
                        break;
                    case "Nokia_Asha_SDK_1_0": //New Asha
                        SetIcon("/images/logo3.png");
                        break;
                    default:
                        break;
                }
            }
        }


        public static void SetIcon(string iconPath)
        {
            string newFileContents = "";

            using (StreamReader sr = new StreamReader("Application Descriptor"))
            {
                string prefix = "MIDlet-1: ";
                string[] midletTokens = null;

                string line = sr.ReadLine();
                while (line != null)
                {
                    if (line.StartsWith(prefix))
                    {
                        line = line.Substring(prefix.Length);
                        midletTokens = line.Split(',');
                        newFileContents += prefix + String.Format("{0},{1},{2}",
midletTokens[0], iconPath, midletTokens[2]);
                    }
                    else
                    {
                        newFileContents += line;
                    }
                    newFileContents += "\n";

                    line = sr.ReadLine();
                }
            }

            using (StreamWriter sw = new StreamWriter("Application Descriptor",false))
            {
                sw.Write(newFileContents.Trim());
            }
        }

    }
}
```

Running the program at build time



- Right click your project in Eclipse, click “Properties” and select “Builders”.
- Click “New…” and select “Program”.
- Give the builder a name. On the “Location” field set the .exe we previously created. Now set your project’s home directory as the “Working Directory”. Click OK.

That’s it! Now every time we build our project, the program we created will be called to make the required changes for us.

Related sources

This article is a complement for the Creating applications for multiple Nokia Java ME platforms article.