A tutorial of how to use Microsoft CMAB

Recently I worked on a .net project, which requires some form of configuration information and needs to store and manipulate the configuration data. In order to resolve this problem, I spent some time exploring the Microsoft Configuration Management Application Block (CMAB). This block provides a more sophisticated approach to configuration management than the inherent ConfigurationSettings capabilities in .Net class library.

This tutorial shows you how to read and write an application configuration file using CMAB according to my experiences. I hope it's helpful for you.

At first, you should perform the following steps in order to use it.
1. Download it from MSDN, then install it according to the instructions.
2. Compile the sample solution to get the useful DLL files, such as Microsoft.ApplicationBlocks.ConfigurationManagment and Microsoft.ApplicationBlocks.ConfigurationManagement.Interfaces.
3. Set a reference to the assembly Microsoft.ApplicationBlocks.ConfigurationManagement.dll in your own project.
4. Add a using statement at the top of your application source files to reference the above DLL file. For example,
Microsoft.ApplicationBlocks.ConfigurationManagement;

Configure app.config file

1. Declare the CMAB configuration section

```
<configuration>
  <configSections>
    <section name="applicationConfigurationManagement"
type="Microsoft.ApplicationBlocks.ConfigurationManagement.ConfigurationManagerSectionHandler,Microsoft.ApplicationBlocks.ConfigurationManagement,
Version=1.0.0.0,Culture=neutral,PublicKeyToken=null" />
  </configSections>
......
</configuration>
```

The above declaration specifies that the CMAB settings are located in a configuration section named applicationConfigurationManagment and the ConfigurationMananagerSectionHandler class is responsible for parsing the configuration settings.

2. Declare the Configuration Section Handler

The CMAB configuration section contains a set of child <configSection> elements. Each <configSection> element contains the declaration of a configuration section.
For example, the CMAB configuration section named applicationConfigurationManagment lists below. In addition, the configuration file excerpt shows that ProxyServerSection section uses the XML storage provider.

```
 <applicationConfigurationManagement defaultSection="ProxyServerSection" >
   <configSection name="ProxyServerSection">
   <configProvider
assembly="Microsoft.ApplicationBlocks.ConfigurationManagement,Version=1.0.
0.0,Culture=neutral,PublicKeyToken=null"
        type="Microsoft.ApplicationBlocks.ConfigurationManagement.Storage.Xml
FileStorage"
        path="YourApplication.exe.config"
        signed="false"
        encrypted="false"/>
   </configSection>
 </applicationConfigurationManagement>
```

From the above declaration, we can see that the CMAB configuration section contains a <configSection> element named **ProxySeverSection**.
*
The path attribute specifies the name of the file in which the configuration section data is stored. The default value is the current application's configuration file.
Please make sure that XML file must exist because XML storage provider will not create a new XML file for you. In the meanwhile, that XML file must contain a single root level element named <configuration>, which must have a child element with the same name as the configuration section - <ProxyServerSection>.

Just like the previous declaration for the CMAB configuration section, We need add a new declaration statement for ProxyServerSection section. The following code is a sample.

```
<configuration>
  <configSections>
   <section name="exceptionManagement"
        type="Rickie.ExceptionManagement.ExceptionManagerSectionHandler,Ric
kie.ExceptionManagement" />
     <!-- Section Handler for CMAB -->
```

```
    <section name="applicationConfigurationManagement"
       type="Microsoft.ApplicationBlocks.ConfigurationManagement.ConfigurationManagerSectionHandler,Microsoft.ApplicationBlocks.ConfigurationManagement, Version=1.0.0.0,Culture=neutral,PublicKeyToken=null" />
       <!—Section Handler for ProxyServerSection -->
       <section name="ProxyServerSection"
       type="Microsoft.ApplicationBlocks.ConfigurationManagement.XmlHashtableSectionHandler,Microsoft.ApplicationBlocks.ConfigurationManagement,Version=1.0.0.0,Culture=neutral,PublicKeyToken=null"/>
 </configSections>
……
</configuration>
```

Note: The default section must always be a Hashtable-based configuration section. So **ProxyServerSection** uses the Hashtable Section Handler implemented by the XmlHashtableSectionHandler class.

3. Declare configuration section used to store configuration settings data

In this tutorial, We use the same application's configuration file to store configuration settings data, YourApplication.exe.Config based on the earlier example.

The following XML listing shows the contents of <ProxyServerSection>. As mentioned early, we put this section in the application configuration file. So we don't need to create a new XML file to store these data.

```
<configuration>
……
  <ProxyServerSection>
     <XmlSerializableHashtable
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <Entries>
       <Entry>
         <key xsi:type="xsd:string">UseProxyServer</key>
         <value xsi:type="xsd:string">false</value>
       </Entry>
       <Entry>
         <key xsi:type="xsd:string">ProxyServer</key>
         <value xsi:type="xsd:string"></value>
       </Entry>
       <Entry>
         <key xsi:type="xsd:string">Port</key>
     <value xsi:type="xsd:string"></value>
    </Entry>
    <Entry>
     <key xsi:type="xsd:string">Username</key>
     <value xsi:type="xsd:string"></value>
    </Entry>
    <Entry>
     <key xsi:type="xsd:string">Password</key>
     <value xsi:type="xsd:string"></value>
    </Entry>
    <Entry>
     <key xsi:type="xsd:string">Domain</key>
     <value xsi:type="xsd:string"></value>
    </Entry>
   </Entries>
  </XmlSerializableHashtable>
 </ProxyServerSection>
......
</configuration>
```

OK. We've finished with configuration file settings.

Read and Write Configuration Section Using the ConfigurationManager Class

Let's start to access application configuration data by ConfigurationManager class which implements the static methods named Read and Write.

1. Read configuration section

```
Hashtable proxyInfo;
// Read the configuration section named ProxyServerSection based on earlier example
proxyInfo = (Hashtable)ConfigurationManager.Read("ProxyServerSection");
```

2. Write configuration section

```
Hashtable proxyInfo;

proxyInfo["UseProxyServer"] = chkUseProxyServer.Checked? "true":"false";
proxyInfo["ProxyServer"] = txtProxyServer.Text.Trim();
proxyInfo["Port"] = txtPort.Text.Trim();
proxyInfo["Username"] = txtUsername.Text.Trim();
proxyInfo["Password"] = txtPassword.Text.Trim();
proxyInfo["Domain"] = txtDomain.Text.Trim();
// Write the configuration section named ProxyServerSection
```

```
ConfigurationManager.Write("ProxyServerSection", proxyInfo);
```

OK. That’s all. This tutorial is simple and just show you how to use a basic function of Microsoft Configuration Management Application Block.