



A tutorial of how to use Microsoft CMAB, Part 2

-- Develop custom configuration section handler

The previous article, *A tutorial of how to use Microsoft CMAB*, shows how to use XmlHashtableSectionHandler provided by CMAB to access application's configuration file. The Hashtable section handler provides support for configuration sections based on the standard .Net Hashtable class. It's simple and fast to implement the required function.

This article will show you how to develop custom configuration section handler. To create a read/write custom section handler, you must create a class that implements the IConfigurationSectionHandlerWriter interface, which is defined in the Configuration Management Application Block and extends IConfigurationSectionHandler.

For example,

```
public class ProxySectionHandler : IConfigurationSectionHandler,  
                                IConfigurationSectionHandlerWriter
```

I've built a sample solution in C# to demonstrate it, which mainly contains three files involved in custom configuration section.

1. App.config

This file is used to generate the application configuration file, which contains the declarations of CMAB configuration section and custom configuration section. The following listing shows sample codes.

```
<?xml version="1.0" encoding="utf-8" ?>  
<configuration>  
  <configSections>  
    <section  
      name="applicationConfigurationManagement"  
      type="Microsoft.ApplicationBlocks.ConfigurationManagement.ConfigurationManagerSectionHandler  
,Microsoft.ApplicationBlocks.ConfigurationManagement,  
Version=1.0.0.0,Culture=neutral,PublicKeyToken=null" />  
  
    <section  
      name="ProxyServerSection"  
      type="CustomConfigurationSectionHandler.ProxySectionHandler,  
CustomConfigurationSectionHandler, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />  
  </configSections>  
  
  <applicationConfigurationManagement defaultSection="ProxyServerSection">  
    <configSection name="ProxyServerSection">  
      <configProvider  
assembly="Microsoft.ApplicationBlocks.ConfigurationManagement,Version=1.0.0.0,Culture=neutral,Public  
KeyToken=null"  
      type="Microsoft.ApplicationBlocks.ConfigurationManagement.Storage.XmlFileStorage"
```

```
        signed="false"
        refreshOnChange="false"
        encrypted="false" />
    </configSection>
</applicationConfigurationManagement>

<ProxyServerSection>
  <ProxyConfigurationData xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <UseProxyServer>true</UseProxyServer>
    <ProxyServer>firewallName</ProxyServer>
    <Port>1234</Port>
    <Username>username</Username>
    <Password>secret</Password>
    <Domain>Domain_name</Domain>
  </ProxyConfigurationData>
</ProxyServerSection>
</configuration>
```

Please pay an attention to these highlighted words, which will be different in the various applications.

2. ProxySectionHandler.cs

This file defines the custom section handler, and is responsible to transform a configuration section back and forth between its in-memory structure and an XmlNode.

ProxySectionHandler
-_xs:XmlSerializer
+ProxySectionHandler +Serialize:XmlNode +Create:object

ReadOnlyProxySectionHandler
-_xs:XmlSerializer
+ReadOnlyProxySectionHandler +Create:object

Notice that ReadOnlyProxySectionHandler class only provides the read functionality from the configuration file.

3. ProxyConfigurationData.cs

This file is used to store the configuration settings while reading or writing configuration data.

4. Read/Write Configuration Data

// Read configuration data from configuration file, then store it into an instance object

```
ProxyConfigurationData proxyData =
(ProxyConfigurationData)ConfigurationManager.Read("ProxyServerSection");
if(proxyData == null)
    throw new ApplicationException("No config on provider.");
```

// Write configuration data into an Xml configuration file by an instance object

```
ProxyConfigurationData proxyData = new ProxyConfigurationData();  
// Assign values to the object public properties  
.....  
ConfigurationManager.Write("ProxyServerSection", proxyData);
```

OK. That's all. This tutorial shows you how to develop a custom configuration section handler to access configuration data.

This document is provided "AS IS" with no warranties, and confers no rights.