Attributes in C#

Authored by : Sushant Banerjee
Email : sushantba@cybage.com

Presented by : Sushant Banerjee
Extn. 7210

Agenda

- Introduction
- Using Attributes
- Custom Attributes

What are Attributes

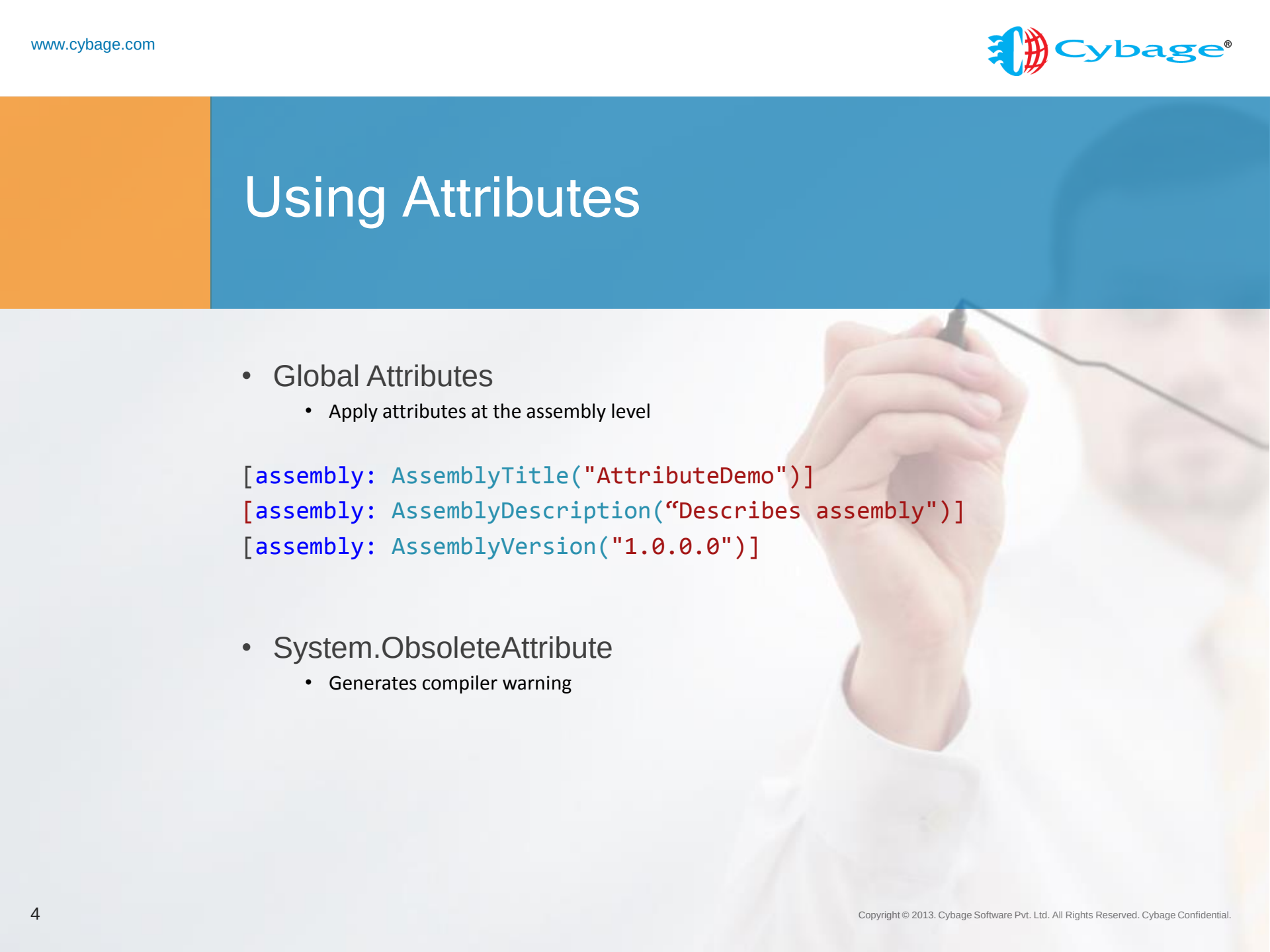
- An assembly contains your code converted into MSIL
- An assembly also contains metadata about your code
- Attributes can add extra information to the metadata
- The tool ildasm.exe allows you to look inside the assembly

Using Attributes

- Global Attributes
  - Apply attributes at the assembly level

```
[assembly: AssemblyTitle("AttributeDemo")]
[assembly: AssemblyDescription(“Describes assembly")]
[assembly: AssemblyVersion("1.0.0.0")]
```

- System.ObsoleteAttribute
  - Generates compiler warning



Custom Attributes

- Creating your own attribute
  - Derive from System.Attribute base class.
  - Add suffix “Attribute” (optional)
- Finally decorate your class adding
  - Attribute Parameters
  - Attribute Targets
  - AttributeUsage

Attribute Parameters

- **Positional Parameter**
  - A position parameter is mandatory
  - it should come before any named parameters.
- **Named Parameter**
  - A named parameter is optional
  - Can be specified in any order
  - Its hould come after all positional parameters.

Attribute Targets

- The target of an attribute is the entity to which the attribute applies.
- By default an attribute applies to the element that it precedes.
- An attribute may target
  - Assembly
  - Field
  - Event
  - Method
  - Class
  - Struct
  - Enum
  - Delegate
  - property etc.

AttributeUsage

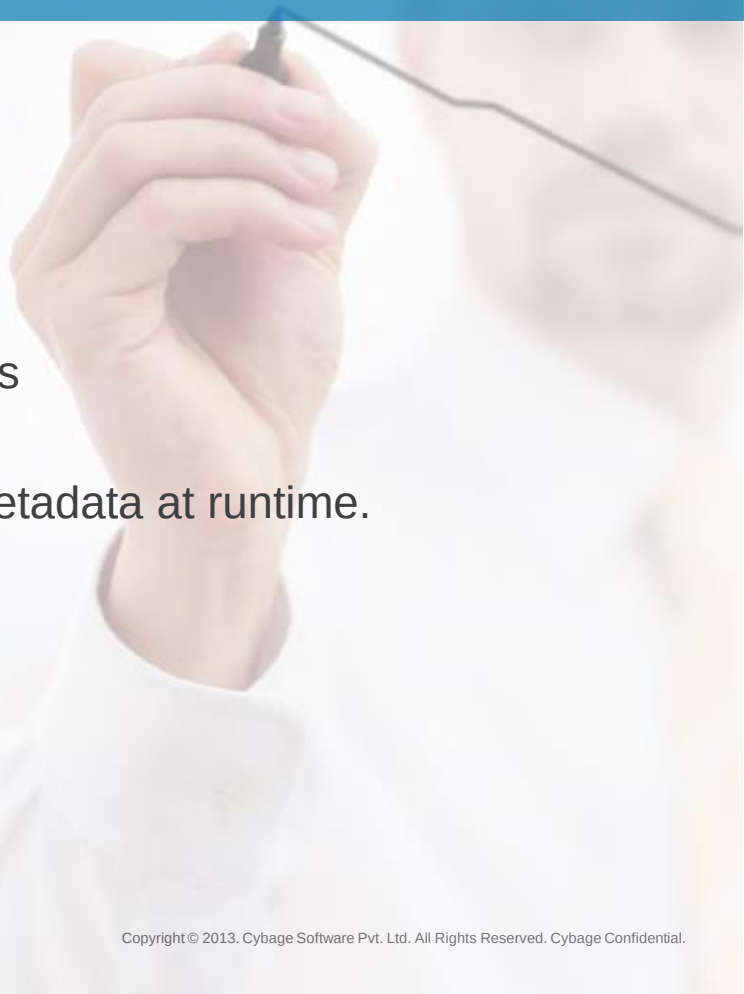
- AttributeUsage is an attribute that can be used to determine how a custom attribute class can be used.

- The default settings of AttributeUsage is as follows:

```
[System.AttributeUsage(System.AttributeTargets.All,
                       AllowMultiple = false,
                       Inherited = true)]
class NewAttribute : System.Attribute { }
```

Using Reflection

- System.Reflection namespace allows
  - Allows you to access types at runtime
  - Create objects at runtime
  - Access members of a type at runtime
- Once metadata added using attributes
- Reflection can be used to read the metadata at runtime.



Serialization

Agenda

- Introduction
- Xml Serialization
- Binary Serialization

What is Serialization

- Serialization is the process of converting an object into a stream of bytes.
- This stream of bytes can be stored permanently in database, file or memory.
- This stream of bytes not only contains just data but information about object's type, version, culture and assembly name.
- The main purpose of serialization is to save the state of an object and recreate it when needed.
- The reverse process of recreating object from stream of bytes is known as **deserialization**.

Uses for Serialization

- Using serialization a developer can perform following actions :
- Sending the object to a remote application using Web Service.
- Passing an object from one domain to another.
- Passing an object through firewall as an XML string.
- Maintaining security or user specific information across applications.

Creating Serializable Object

- **System.Runtime.Serialization** provides classes for serializing and deserializing objects.
- You need to apply an attribute **SerializableAttribute** to a type to indicate that instances of this type can be serialized.
- If you do not want a field within your class to be serializable, apply the attribute **NonSerializedAttribute**.

Types of Serialization

- Serialization can be following types :
- **Binary Serialization** : Binary serialization uses binary encoding and saves objects state in binary format.
- **XML Serialization** : XML Serialization serializes the object into an XML stream and saves the objects state in xml format.

Binary Vs. XML Serialization

- **System.Runtime.Serialization.Formatters.Binary** provides all the necessary classes for serializing and deserializing objects in binary format.
- You need to use **BinaryFormatter** class for binary serialization and deserialization.

- **System.Xml.Serialization** contains the classes necessary for serializing and deserializing objects in XML format.
- You need to use **XmlSerializer** class for xml serialization and deserialization.

Bibliography, Important Links

- http://msdn.microsoft.com/en-IN/library/z0w1kczw.aspx
- http://msdn.microsoft.com/en-IN/library/ms233843.aspx

Any Questions?

Thank you!