Module 14: Attributs

Vue d'ensemble

- Aperçu des caractéristiques
- Définition d'attributs personnalisés
- Récupération des valeurs d'attribut

Aperçu des caractéristiques

- Introduction à Attributs
- Application des attributs
- Attributs prédéfinis communes
- Utilisation de l'attribut sous condition
- Utilisation de l'attribut DllImport
- Utilisation de l'attribut de transaction

Introduction à Attributs

- Les attributs sont:
  - Balises déclaratives qui transmettent des informations à l'exécution
  - Stockée avec les métadonnées de l'élément
- . NET Framework fournit des attributs prédéfinis
  - Le runtime contient le code pour examiner les valeurs d'attributs et d'agir sur eux

Application des attributs

- Syntaxe: Utilisez des crochets pour spécifier un attribut

```
[attribut(position_paramètres, named_parameter = valeur, ...)]
élément
```

- P
  pouvez:
  - Spécifier plusieurs attributs entre crochets séparés
  - Utilisez un crochet unique et attributs séparés par des virgules
  - Pour certains éléments, tels que des ensembles, préciser le nom de l'élément associé à l'attribut explicitement

Attributs prédéfinis communes

- . NET fournit de nombreux attributs prédéfinis
  - Caractéristiques générales
  - attributs d'interopérabilité COM
  - La manipulation des attributs de transaction
  - Attributs de construction de composants de concepteur visuel

Utilisation de l'attribut sous condition

- Sert d'outil de débogage
  - Provoque la compilation conditionnelle d'appels de méthode, en fonction de la valeur d'un symbole définie par le programmeur
  - Ne provoque pas de compilation conditionnelle de la méthode elle-même

```
using System.Diagnostics;
...
class MyClass
{
  [Condition ("Test")]
  LaMéthode public static void ()
  {
    ...
  }
}
```

- Restrictions sur les méthodes
  - Doit avoir le type de retour **vide**
  - Ne doit pas être déclaré comme **override**
  - Ne doit pas être d'une interface héritée

technofutur TIC

WAVENET Intelligent IT Solutions

Utilisation de l'attribut DllImport

- Avec l'attribut DllImport, vous pouvez:
  - Appelez code non managé dans les DLL d'un environnement C #
  - Marquer une méthode externe pour montrer qu'il réside dans une DLL non managée

```
utilisant System.Runtime.InteropServices;
...
public class MyClass ()
{
  [DllImport ("MYDLL.DLL", EntryPoint = "MyFunction")]
  public static extern int MyFunction (string param1);
  ...
  int result = MyFunction ("Bonjour le code non managé»);
  ...
}
```

technofutur TIC

WAVENET Intelligent IT Solutions

Utilisation de l'attribut de transaction

- Pour gérer les transactions dans COM +
  - Spécifiez que votre appareil soit inclus lors d'une opération de validation est demandée
  - Utilisez un attribut de transaction sur la classe qui implémente le composant

```
utilisant System.EnterpriseServices;
...
[Transaction (TransactionOption.Required)]
MyTransactionalComponent public class
{
  ...
}
```


Définition d'attributs personnalisés

- Définir attribut personnalisé Portée
- Définition d'une classe d'attributs
- Traitement d'un attribut personnalisé
- Utilisation des attributs multiples

Définir attribut personnalisé Portée

- Utilisez la balise AttributeUsage à définir la portée
  - Exemple

```
[AttributeUsage (AttributeTargets.Method)]
MyAttribute public class: System.Attribute
{...}
```

- Utilisez l'« bit »ou« opérateur (|) pour spécifier plusieurs éléments
  - Exemple

```
[AttributeUsage (AttributeTargets.Class | AttributeTargets.Struct)]
MyAttribute public class: System.Attribute
{...}
```

technofutur TIC

WAVENET Intelligent IT Solutions

Définition d'une classe d'attributs

- Issu d'une classe d'attributs
  - Toutes les classes d'attributs doivent dériver de System.Attribute, directement ou indirectement
  - Suffixe de nom de classe de l'attribut avec "Attribut"
- Composants d'une classe d'attributs
  - Définir un constructeur unique pour chaque classe d'attributs en utilisant un paramètre de position
  - Utiliser les propriétés de définir une valeur en option en utilisant un paramètre nommé

technofutur TIC

WAVENET Intelligent IT Solutions

Traitement d'un attribut personnalisé

Le processus de compilation

Une. Recherches pour la classe d'attributs

2. Elle vérifie le champ d'application de l'attribut

3. Vérifie un constructeur dans l'attribut

4. Crée une instance de l'objet

5. Vérifie un paramètre nommé

6. domaine des jeux ou des biens de valeur de paramètre nommé

7. Enregistre l'état actuel de la classe de l'attribut

technofutur TIC

WAVENET
Intelligent IT Solutions

Utilisation des attributs multiples

- Un élément peut avoir plus d'un attribut
  - Définir les attributs séparément
- Un élément peut avoir plus d'une instance de la même attribut
  - Utilisez AllowMultiple = true

technofutur TIC

WAVENET
Intelligent IT Solutions

Récupération des valeurs d'attribut

- Examen classe de métadonnées
- Interrogation d'information d'attribut

Examen classe de métadonnées

- Pour interroger les informations de métadonnées de classe:
  - Utilisez la classe MemberInfo dans System.Reflection
  - Remplir un objet MemberInfo en utilisant System.Type
  - Créer un objet System.Type en utilisant l'opérateur typeof
- Exemple

```
System.Reflection.MemberInfo typeInfo;
typeInfo = typeof (MyClass);
```

Interrogation d'information d'attribut

- Pour récupérer des informations d'attribut:
  - Utilisez **GetCustomAttributes** pour récupérer toutes les informations de l'attribut comme un tableau

```
System.Reflection.MemberInfo typeInfo;
typeInfo = typeof (MyClass);
objet [] attrs = typeInfo.GetCustomAttributes (false);
```

  - Parcourir le tableau et à examiner les valeurs de chaque élément dans le tableau
  - Utilisez l' **IsDefined** Procédé pour déterminer si un attribut particulier est défini pour une classe

Lab 14.1: Définition et utilisation des attributs

Examen

- Aperçu des caractéristiques
- Définition d'attributs personnalisés
- Récupération des valeurs d'attribut

technofuturTIC
centre de compétences

WAVENET
Intelligent IT Solutions