Lokalisierung mit dem .NET Framework

Herfried K. Wagner

Microsoft MVP

Überblick

- Lokalisierung und Internationalisierung
- Ressourcendateitypen
- Werkzeuge zur Lokalisierung
- Zugriff auf Ressourcen
- Lokalisierung mit Satellitenassemblies
- Win32-Ressourcen in .NET

Lokalisierung und Internationalisierung

- Lokalisierung:

  Prozeß des Anpassens von Ressourcen an eine bestimmte Sprache oder Kultur (Symbole, Texte, Klänge, Farben, Maßeinheiten, rechtliche Bestimmungen, Position der Steuerelemente, RTL-Anordnung und -Schreibrichtung)

- Internationalisierung:

  Anpassen des Programms an verschiedene Sprachen und Kulturen ohne Änderung des Quellcodes (Eingaben, Ausgaben, Darstellung)

Werkzeuge für die Lokalisierung

- Microsoft Visual Studio ab Version 2002, Microsoft Visual Studio 2005 Express: Lokalisierung von Ressourcen und Formularen
- Windows Forms Resource Editor (Winres): Lokalisierung von Formularen
- Resource Refactoring Tool (http://www.codeplex.com/ResourceRefactoring): Extraktion von Zeichenfolgen in Ressourcen
- Resourcer for .NET (http://www.aisto.com/roeder/dotnet/): Bearbeiten von Ressourcendateien
- Resource File Generator (Resgen)

Ressourcendateitypen

- ResX: XML-basiertes Ressourcenformat
- TXT: INI-ähnliches Textformat für Zeichenfolgen
- „.resources“: Binäres Ressourcenformat:
  - Mit Resgen aus ResX oder TXT erstellt
  - Kann in Assemblies eingebettet werden
- Erstellen von Code zur typsicheren Kapselung des Ressourcenzugriffs:
  - Resgen
  - Klasse `StronglyTypedResourceBuilder`

Lokalisierung von Windows Forms

- Eigenschaft `RightToLeft`: Darstellung des Steuerelements (Text etc.) von rechts nach links
- Eigenschaft `RightToLeftLayout`: Anordnen der Steuerelemente von rechts nach links
- Eigenschaft `Localizable`: Gibt an, ob das Formular lokalisiert wird
- Eigenschaft `Language`: Ändern der Sprache zum Anlegen und Bearbeiten von Ressourcen verschiedener Kulturen zur Entwurfszeit

Beispiel 1: Lokalisieren eines Formulars mit Visual Studio

Windows Forms Resource Editor

- Einsatz vorwiegend zur Lokalisierung von Formularen in Übersetzungsagenturen
- Teil des .NET Framework SDKs
- Inkompatible Ressourcenformate zwischen Visual Studio .NET 2003 und Winres (behoben mit Visual Studio 2005)
- Eigenschaft `Localizable` zu lokalisierender Formulare muß auf `True` gesetzt sein
- Alternative: Visual Studio 2005 Express (kostenlos, gewerblicher Einsatz gestattet)

Beispiel 2: Lokalisieren eines Formulars mit dem Windows Forms Resource Editor

Ändern der Sprache zur Laufzeit

- Nicht vorgesehen
- Status der Anwendung (z.B. zur Laufzeit geänderte Beschriftungen) müßte berücksichtigt werden
- Meist nicht erforderlich
- Stattdessen Änderung bei Neustart der Anwendung

Lokalisieren von „Resources.resx“

- Keine direkte Möglichkeit innerhalb von Visual Studio 2005 und dem Windows Forms Resource Editor
- Kopieren der „Resources.resx“
- Umbenennen in „Resources.*Sprachbezeichner*.resx“ (z.B. „Resources.de.resx“, „Resources.en-US.resx“)
- Hinzufügen der umbenannten Datei zum Visual Studio-Projekt im Projektmappenexplorer
- Lokalisieren der Ressourcen

Ressourcenzugriff mit `My.Resources`

- Typsicherer und komfortabler Zugriff auf Ressourcen zur Laufzeit, z.B.
  ```
  Me.PictureBox1.Image =
  My.Resources.WizardFinish
  ```
- Ressourcen werden unter „My Project“ → „Ressourcen“ oder in dazugehörigen lokalisierten ResX-Dateien angelegt
- Bei lokalisiertem `My.Resources` wird die Ressource anhand der aktuellen Kultur oder dem Wert der Eigenschaft `Culture` gewählt

Beispiel 3: Lokalisieren von „Resources.resx“ und Zugriff über `My.Resources`

Ressourcenzugriff mit dem `ResourceManager`

- Nicht typsicherer Zugriff auf Ressourcen verschiedener Kulturen zur Laufzeit
- `My.Resources` kapselt `ResourceManager`
- Mehr Möglichkeiten als `My.Resources`
- Optionales Zurückfallen, falls die gewünschte Ressource für die Kultur nicht existiert
- Serialisierung von Ressourcen
- Erweiterte Ladeoptionen, z.B. Laden aus „.resources“-Dateien

Auswahl der Kultur

- Anwendung startet mit der Kultur des Benutzers
- Kulturen werden auf Threadbasis festgelegt
- Eigenschaften `Thread.CurrentCulture`, `Application.CurrentCulture`: Kultur des Threads, z.B. zum Formatieren verwendet
- Eigenschaft `Thread.CurrentUICulture`: Zum Laden von Ressourcen benutzte Kultur
- Manche Methoden erlauben die Übergabe eines `CultureInfo`-Objektes zur Steuerung der Formatierung, z.B. `ToString`

Zugriff auf Ressourcen über `CultureInfo`

- Stellt Information über eine bestimmte Kultur bereit:
  - Kulturspezifische Bezeichnungen
  - Schriftsystem
  - Kalender
  - Formatierung von Datumsangaben
  - Sortierung von Zeichenfolgen
- Eigenschaft `InvariantCulture`: Geeignet für kulturneutrale Ein- und Ausgabe
- Namensraum `System.Globalization`

Lokalisierung mit Satellitenassemblies

- Ähnliches Konzept bereits in Win32
- Hauptassembly enthält Standardressourcen
  - Angabe der Kultur der Standardressourcen über das Attribut `NeutralResourcesLanguageAttribute`
  - Zurückfallen auf Ressourcen eines Satelliten möglich
- Satellitenassemblies enthalten jeweils Ressourcen für eine bestimmte Kultur

Versionierung mit Satellitenassemblies

- Hauptassembly ändert sich, Satellitenassemblies bleiben unverändert:

  Attribut `SatelliteContractVersionAttribute` gibt an, mit welcher Version der Satellitenassemblies das Hauptassembly zusammenarbeiten kann

- Satellitenassemblies werden verändert, Hauptassembly bleibt gleich:

  Über ein Policy-Assembly im GAC kann zur neuen Version der Satellitenassemblies gebunden werden

Benennung von Satellitenassemblies

- Satellitenassemblies liegen in einem Unterverzeichnis des Verzeichnisses der Anwendung
- Verzeichnisname ist der Sprachbezeichner (z.B. „en-US“ oder „de“)
- Dateiname der Satelliten ist immer „*Hauptassemblyname*.resources.dll“

Erstellen von Satellitenassemblies ohne Visual Studio

- Erforderlich, wenn Ressourcen außer Haus lokalisiert oder Satellitenassemblies nachträglich verändert werden
  - Erstellen von „.resources“-Dateien aus den ResX-Dateien mit Resgen
  - Linken und ggf. Signieren des Ressourcenassemblies mit „Al.exe“ unter Angabe der „.resources“-Datei

Signieren von Satellitenassemblies

- Satellitenassemblies müssen mit demselben privaten Schlüssel wie das Hauptassembly signiert werden
- Private Schlüssel sind aus Sicherheitsgründen in Unternehmen meist nur einigen Mitarbeitern zugänglich
- Verzögertes Signieren während der Entwicklung möglich

Beispiel 4: Lokalisierung einer Anwendung mit Satellitenassemblies

Lokalisierung mit „.resources“-Dateien

- Anstelle von Satellitenassemblies
- Erstellen des `ResourceManager`-Objektes über die Methode `CreateFileBasedResourceManager`
- Einsatz vorwiegend zu Testzwecken, da „.resources“-Dateien nach Zugriff gesperrt bleiben
- Keine Versionierung möglich

Win32-Ressourcen in .NET

- Windows nutzt weiterhin Win32-Ressourcen:
  - Anwendungs- und Verknüpfungssymbole
  - Zeichenfolgen für die Windows Shell
  - Protokoll `res:` zum Zugriff auf Ressourcen
  - Manifestressourcen (`RT_MANIFEST`)
- Migration von Win32 (z.B. Visual Basic 6.0) zu .NET

Die ResLib-Bibliothek

- Kapselung der wichtigsten Win32-Ressourcenfunktionen
- Zugriff über PInvoke mit Win32-API
- Klassen `ResourceManager` und `ResourceUpdater`
- Beispielanwendung: ThemeUtil zum Einbetten von Manifestressourcen in Binärdateien

Besten Dank für die Aufmerksamkeit!