PL/B And .NET

Bill Keech

Sunbelt Computer Systems

OVERVIEW

- .NET
- PL/B .NET Interaction
- Basic .NET Object
- Creating .NET Objects
- Using .NET Events
- Standard .NET Types
- Static .NET Objects
- .NET Types
- .NET Enumerations
- .NET Arrays
- Sample .NET Code

.NET

- General language runtime
  - Common Language Runtime (CLR)
- .NET Framework
  - Provides a large object class library
- Programs are stored as p-code
- Execution is done by a Just In Time compiler
  - Compiles p-code to native machine instructions

.NET

- Common development environment
- Common Type System (CTS)
  - Replaces VARIANTs
- All languages must use the CTS
- Assemblies are binary components that are managed and run by the Common Language Runtime

PL/B .NET Interaction

- Minimum .NET framework version 2.0
- New controls require .NET framework version 3.0
- Must use PLBNET or PLBCLINET command
- Runtime is CLR code
- Runtime bridges managed and unmanaged world

PL/B .NET Interaction

- WINAPI still works
  - Runtime uses the It Just Works (IJW) interface to get to WIN32 API
- Automation support still works
- NETOBJECT variable used to represent a .NET class
- Late binding is used by the runtime

PL/B .NET Interaction

- Supports PL/B Instructions
  - CREATE
  - DESTROY
  - GETPROP
  - SETPROP
  - EVENTREG
  - Methods

- PL/B programs can be incrementally enhanced

Basic .NET Object

- Every data variable in .NET is an object
- All objects are referenced by namespace
- Names are case sensitive
- Assemblies are the actual code files on disk

Basic .NET Object

- The ultimate base class of all classes in .NET
  - Equals
    - Determines whether two Object instances are equal.
  - GetHashCode
    - Serves as a hash function for a particular type.
  - GetType
    - Gets the Type of the current instance.

Basic .NET Object

- The ultimate base class of all classes in .NET
  - ReferenceEquals
    - Determines whether the specified Object instances are the same instance.
  - ToString
    - Returns a String that represents the current Object.

Basic .NET Object

- Use Dispose method to cleanup resources

Creating .NET Objects

- CREATE of a NETOBJECT invokes it's constructor
- CLASS specifies the actual .NET object
- CLASS starting with a ‘!’ means client side create
- ASSEMBLY specifies the .dll the code is in
- *$ specifies any need parameters
  - Must be in specific parameter order
  - Parameters used to pick proper constructor
- Any property can be added to CREATE

Creating .NET Objects

```
Create    Color:
          Class="System.Drawing.Color”:
          Assembly="System.drawing"

Create    FI:
          Class="System.IO.FileInfo”:
          Assembly="mscorlib”:
          *$="c:\netx.pls"
```

Using .NET Events

- Events handled by EventRegister
- Object events are by name, not number
- Internal events using EVENTSEND are still by number
- ARG1 contains the object that caused the event
- ARG2 contains the event specific data
- ARG1, and ARG2 must be NETOBJECTs

Using .NET Events

```
Info                NetObject
Data                Dim 10
    EventRegister ListBox:
                              "ItemCheck”:
                              TestItem:
                              ARG2=Info
TestItem
    GetProp Info,*Index=Data
    Display “Index is: ",Data
    Return
```

Standard .NET Types

- PL/B data types are converted to/from .NET types

- Some examples are
  - DIM System.String
  - INTEGER (1 byte) System.Byte
  - VARIANT (VT_BOOL) System.Boolean
  - FORM (with decimal pt) System.Double
  - FONT objects can be converted to .NET fonts
  - COLOR objects can be converted to .NET colors

- Basic .NET types can be created with an initial value

Standard .NET Types

- Some examples are:

```
Create NetString:
          Class="System.String”:
          Assembly="mscorelib”:
          *$=DIM20

Create NetVal:
          Class="System.Int32”:
          Assembly="mscorelib”:
          *$=Form8
```

Static .NET Objects

- Some .NET object do not get created but already exist
- CLASS must be terminated by a ‘;’ to indicate a static object
- System.IO.File class provides static methods
  - These methods are create, copy, delete, move, and open files

Static .NET Objects

```
Create FileObj:
        CLASS="System.IO.File;":
        Assembly="mscorlib"

FileObj.OpenWrite Giving FS1 Using "c:\netsoap1.soap"

FileObj.Delete("c:\test1.txt");

FileObj.Copy("c:\test2.txt", "c:\test1.txt");
```

.Net Types

- All .NET objects have type information
- The GetType method can be used to obtain the type

```
Create Color:
        Class="System.Drawing.Color”:
        Assembly="System.drawing"

Color.GetType Giving ColorType
```

.Net Types

- Types can also be obtained without creating an instance
- The CLASS specification must end in “;type”

Create ColorType:

```
        Class="System.Drawing.Color;type”:
        Assembly="System.drawing"
```

.NET Enumerations

- Enumeration is a special form of value type
- Derives from System.Enum
- Used to allow a string literal to represent a constant value
- Enumerations are specified by namespace, a colon, and the enumeration

.NET Enumerations

Create FileMode:

```
        CLASS="System.IO.FileMode:Create":
        Assembly="mscorlib"
```

Create FS:

```
        CLASS="System.IO.FileStream":
        Assembly="mscorlib":
        *$="c:\netsoap.soap":
        *$=FileMode
```

.NET Arrays

- Arrays are really collections of .NET objects

- Namespace is System.Collections

- There are many types of collections such as:
- ArrayList
  - Simple re-sizeable, index-based collection
- SortedList
  - Sorted name/value pairs
- HashTable
  - Name/value pairs that allow retrieval by name or index

.NET Arrays

- Length property represents the total number of elements
- Add method is used to add one item
- Remove method removes on item
- Resize method changes the size of an array

.NET Arrays

```
Create  ColorPref:
          CLASS="System.Collections.Hashtable":
          Assembly="mscorlib"

ColorPref.Add Using "Jeff", "Blue"

ColorPref.Add Using "Fred", "Green"

ColorPref.Add Using "Mary", "Red"
```

Sample .NET Code

- Show Sample .NET Program

QUESTIONS?

That's All!!