CONVERSION OF VARIOUS TYPES OF JAVA-LANGUAGE APPLICATIONS

Priyanka yadav[1], Vishal Sharma[2], Priti Yadav[3]

[1]*Computer Science Department, Dronacharya College of Engineering/ Maharishi Dayanand University, India*

[2]*Computer Science Department, Dronacharya College of Engineering/ Maharishi Dayanand University, India*

[3]*Computer Science Department, Dronacharya College of Engineering/ Maharishi Dayanand University, India*

ABSTRACT:-

This paper discuss about the java: object oriented language and conversion of various types of java language applications. Java is a set of several computer software. So it is needed to convert various applications like java beans applications, EJB Applications, JAAS Applications, Java Swing Applications, CORBA Applications,etc.

Keywords:- Beans, CORBA, EJB, JAVA, JAAS.

I. INTRODUCTION

Java is a set of several computer software and specifications developed by Sun Microsystems, later acquired by Oracle Corporation that provides a system for developing application software and deploying it in a cross- platform devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. While less common, Java applets are sometimes used to provide improved and secure functions while browsing the World Wide Web on desktop computers

Writing in the Java programming language is the primary way to produce code that will be deployed as Java Byte Code; byte code compilers are also available for other languages, including Ada, Java Script, Python, and Ruby. Several languages have been designed to run natively on the Java Virtual Machine (JVM), including Scala, Clojure and Groovy. Java syntax borrows heavily from C and C++, but object-oriented features are modeled after

Smalltalk and Objective C.Java eliminates certain low-level constructs such as pointers and has a very simple memory model where every object is allocated on the heap and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the JVM.

A.CONVERSION OF JAVABEANS APPLICATIONS

Before you convert a JavaBeans application, make sure that it has a manifest file. Otherwise, it will be treated like any other file. The manifest file must be named MANIFEST.MF and placed in a folder named META-INF. You can rename the existing file if necessary.

Only JavaBeans identified in the manifest file by the standard convention are converted as such. For example:

Name: x\y class

Java-Bean: True | False

In this example, x represents the package and y the bean class component.

The correct identification of beans affects both the class to which they are converted and the application of information contained in an associated **BeanInfo** class in the conversion process. For example, a **JPanel** object is normally converted to a System. Window. Form.Panel object. However, if it is identified in the manifest file as a JavaBeans, it is converted to a System.Windows. Forms.User Control object.

Visual JavaBeans that inherit from either the **java.awt.Panel** or the **javax.swing.JPanel** class are converted to inherit from the **System.Windows.Forms.UserControl** class.

Information in associated **BeanInfo** classes is applied during the conversion process, but there is no direct equivalent in the .NET Framework to these classes

B CONVERSION OF EJB APPLICATION

WHEN CONVERTING AN ENTERPRISE JAVABEANS (EJB) APPLICATION, ONLY EJB-RELATED CLASS AND INTERFACE FILES ARE INCLUDED IN THE CONVERTED PROJECT. IF YOUR APPLICATION HAS DEPENDENCIES ON OTHER UTILITY CLASSES, YOU MUST ADD THEM TO THE PROJECT MANUALLY. ALL UTILITY CLASSES ARE INCLUDED IN THE CLIENT PROJECT BY DEFAULT AND CAN BE MANUALLY EXCLUDED WHEN NEEDED.

Naming lookup is handled differently in the .NET Framework, resulting in compile errors related to the naming context. Related code can be commented out in the Visual C# files.

You must manually set up component security after conversion.

Since message-driven beans are converted to components, you must verify any code that writes output to the console.

When deploying a component, the DLL must be signed with a strong name. You must generate a key file and integrate it into the assembly manually.

Review the transaction settings for your converted component after deployment.

If your component is published to a Windows 2000 Server, you must remove the **Private Component** tag for all serviced components.

C.CONVERSION OF JAAS APPLICATIONS

The .NET Framework uses role-based security with principal, identity, and permission classes to handle security. You can choose one of the built-in security modules within the .NET Framework, rather than building one yourself. When you convert Java Authentication and Authorization Service (JAAS) applications to the .NET Framework, you must take into account the differences between the two approaches to security.

All JAAS configuration files must be renamed as JAAS. config to be processed by Java Language Conversion Assistant. These are converted to App.config files, which can be used by support-class methods to obtain authentication modules and register them with the authentication manager.

The **LoginContext** class is converted to the static **System.Security.AuthenticationModule** class, which has different behavior.

The **LoginModule** class is converted to the IauthenticationModule interface. In the Java language, the **LoginContext**object registers a **LoginModule** object, which uses callback handlers to request input from the user and login module to authenticate users. In the .NET Framework, authentication modules are registered with the authentication manager, which loops through registered authentication modules to return authorization information.

A generic System.Security.SecurityException is used instead of the different JAAS exceptions.

The **Subject** class is converted to the System.Security.Principal.GenericPrincipal class, which has different behavior. There is also a System.Security.Principal.WindowsPrincipal class for use with Windows authentication.

D.CONVERSION OF JAVA SWING APPLICATIONS

Swing applications are converted to the System.Windows.Forms namespace. Therefore, the following Swing elements have no direct equivalent in Visual C# and cannot be converted.

[1]. The **javax.swing.plaf** package is not supported. There is no equivalent in the .NET Framework to the classes that encapsulate Swing pluggable look-and-feel or methods that refer to it.

[2]. Renderers and editors are not supported.

[3]. Java AWT layouts are not supported. Although these are not Swing-specific, they affect most Swing applications.

The following elements have limited support or require significant manual adaptation after conversion with the Java Language Conversion Assistant:

- Models are only partially supported.
- Classes such as **Jtree** and **Jtable** cannot be converted directly to their equivalents in the .NET Framework because of their dependence on models. Support classes are implemented to help match these classes, but not all items can be mapped.
- The **ContentPane** class and other intermediate-level containers cannot be converted directly. For example, the **Jframe** object contains controls in its content pane, but the .NET Framework System.Windows.Forms.Form class contains controls in itself, without any intermediate object. This results in inheritance issues, and conversion requires significant user input.
- Classes whose behavior is specified by constant values cannot be converted completely automatically. For example, the **File Dialog** class has a constant that specifies whether it opens or saves a file. This is handled by two separate classes in Visual C#.
- Event handling is built on a different model in the .NET Framework, so all Swing events require manual adaptation after conversion. A sample of event handling conversion is included in the javax.swing Sample topic.

E.CONVERSION OF ACTIVEX CONTROLS

The **com.ms.wfc.ax** package contains classes and interfaces for Microsoft ActiveX controls and ActiveX Automation objects.

In the .NET Framework, Windows Forms can host only Windows Forms controls. ActiveX controls must be wrapped to appear like Windows Forms controls in a wrapper class that derives from the System.Windows.Forms.AxHost class.

You can convert ActiveX component in any of the following ways:

- Visual Studio automatically converts COM types in a type library to metadata in an assembly.
- Import type libraries provide command-line switches to adjust metadata and generate an interoperation assembly and a namespace.
- Custom wrappers are the most labor-intensive option, but can be individualized to your application needs.

F.CONVERSION OF CORBA APPLICATIONS

Make sure that no necessary code is included in the code for skeletons or stubs in your source files. This code will be lost in conversion.

The directory that you specify as the target must be included in the CLASSPATH.

CORBA applications are converted to .NET Framework Remoting over HTTP by default. This means that the converted application no longer uses CORBA clients and servers.

.NET Framework Remoting uses direct client/server connections, with no intervening naming service or middle tier. Therefore, there is no object reference broker (ORB).

Much of the code needed to create the CORBA naming service is unnecessary in .NET Framework Remoting. This includes the classes and collections typically packed into a hash table or array of properties.

Comment out all references to classes in the following packages:

A. **org.omg.CosNaming**

B. **org.omg.CosNamingContextPackage**

CORBA applications must first obtain the naming context. Java Language Conversion Assistant treats this as a remote lookup, and this is not needed in .NET Framework Remoting

Conclusion: We have seen in the research that various namespaces are required to convert various applications to the java. Sometimes some applications require direct client/server connections and built-in security modules within the .NET Framework.

II. **CONCLUSIONS**

We have seen in the research that various namespaces are required to convert various applications to the java. Sometimes some applications require direct client/server connections and built-in security modules within the .NET Framework.

REFERENCES

[1]. Wikipidea
[2]. MSDN(WWW.MICROSOFT.COM)
[3]. http://www.artima.com/