Contents

JADE .NET Exposures
- Introduction ....................................................................................................... 4
- DotNetConnection Application ........................................................................ 4
  Exercise 1 – Creating a DotNetConnection Application .................................. 4
- JADE Exposures ............................................................................................... 5
  Exercise 2 – Using the JADE Exposure Wizard ............................................. 6
- Dynamic Link Libraries (*.dll Files) ................................................................ 10
  Exercise 3 – Creating a DLL File ................................................................. 11
  Exercise 4 – Importing JADE DLL Files into .NET ....................................... 14
- C#, XAML, and the Windows Presentation Framework ...................................... 17
  Exercise 5 – Creating a WPF Application ..................................................... 19
JADE .NET Exposures

Introduction

JADE provides for integration with Microsoft .NET systems through use of the JoobContext object.

In this module, the Banking system that is built in the JADE Developer’s course is exposed to .NET, allowing for a WPF front-end application to display and add to the JADE back-end database.

As the Banking system was designed with a separation between the model and the view, the view can be substituted with a completely different view without the need for any change to the model code.

DotNetConnection Application

For Microsoft .NET to use JADE classes, it maintains a reference to a JoobContext object that is generated by JADE and it provides an interface to the JADE classes by delegating requests to the appropriate JADE classes.

As the JoobContext object needs to have a dedicated process to be able to read properties or run methods of JADE classes, we use a non-GUI application called DotNetConnection to generate this process.

Exercise 1 – Creating a DotNetConnection Application

In this exercise, you will create the DotNetConnection application that the JoobContext object will use to delegate requests from .NET to JADE.

1. Select BankingViewSchema in the Schema Browser and then open the Application Browser by pressing Ctrl+L or clicking on the Browse Applications icon in the toolbar.
2. Select the Add command from the Application menu and then fill out the Application sheet of the Define Application dialog as follows.

- Enter DotNetConnection in the Name text box.
- Select Non-GUI in the Application Type drop-down list box.
- Check the Show Super Class Methods check box, so that you can set the initialize method to BankingModelSchema::initialize in the Initialize Method combo box.
- Click OK.

JADE Exposures

Rather than letting external applications have full access to JADE databases, JADE defines an exposure, which contains only the properties and methods that the specific external application requires.
The advantage of this is twofold, as it:

- Provides a more-simple interface to the JADE database
- Allows for greater control over which JADE properties and methods external applications can access

**Exercise 2 – Using the JADE Exposure Wizard**

In this exercise, you will define the exposure that the .NET application will access.

1. Select **BankingViewSchema** in the Schema Browser, and then select the **Exposures** command from the Browse menu.

   Right-click on the **Exposure Browser** and then select the **Add** command, to create a new C# exposure.
2. Name the exposure **BankingClasses** and then select **BankingModelSchema** option in the **Superschemas up to** drop-down list box.

3. Select the following classes.
   a. **Bank**
   b. **BankingModelSchema** (a subclass of the **RootSchemaApp** application class)
   c. **Customer**.

These will be the classes that are accessible from .NET.
4. Select the Bank class and then select (check) the allCustomers property from the features on the right of the sheet. This exposes the allCustomers collection to .NET; the other methods and properties are not exposed.

5. In the BankingModelSchema class, check the myBank feature.
6. In the **Customer** class, check the **address**, **firstNames**, **lastName**, **number**, and **setPropertiesOnCreate** feature properties.

7. As the steps on sheets 4 and 5 of this wizard do not require any changes, click **Next** until you reach step 6 (that is, the **Generate** sheet).

   Fill out this sheet, as follows.
   
a. Enter `C:\Projects\BankingClasses` in the **Output Directory** text box. This is the directory where the C# project file and class files will be created.
   
b. Check the **Generate Sample .csproj File** and **Generate Sample .config File** check boxes.
   
c. Enter the absolute path of the JADE database directory (including the **system** folder) in the **Jade Database Path** text box.
   
d. Enter the location and file name of your JADE initialization file (which is probably called **jade.ini**) in the **Jade Initialization File Path** text box.
e. Check the **Sign-on to Jade as Multiuser** check box.

![JADE.NET Exposures Configuration Wizard](image.jpg)

8. Click **Generate**.

   The wizard does not close, but **Generation completed** should be displayed at the lower left of the wizard. You can now click **Close**.

**Dynamic Link Libraries (*.dll Files)**

A Dynamic Link Library is an external method (or routine) written in any language that can create a Dynamic Link Library (DLL). DLL methods can then be executed as standard JADE methods from within JADE.

The purpose of creating DLLs is to allow different environments to integrate; for example, allowing an application written in C# to use JADE methods.
Exercise 3 – Creating a DLL File

In this exercise, you will use the generated C# project to create the BankingClasses.dll file.

1. In the File Explorer, navigate to C:\Projects\BankingClasses.

Find the Solution Explorer and then select **BankingClasses** under Solution 'BankingClasses'.

3. Press Alt+Enter to open the Properties menu.
Change the Target framework to **.NET Framework 4.7.1**.

4. In the following message box that is then displayed, click **Yes**.
5. Click the **Build** tab, and then change the platform target from **Any CPU** to **x64**.

![Platform Target](image)

6. Build the project by pressing **Ctrl+Shift+B**. You should then see a message like the following displayed in the Output window.

![Output Window](image)

Building this project with the correct settings creates a .dll file for the JADE exposure defined in Exercise 2 of this module.

**Exercise 4 – Importing JADE DLL Files into .NET**

In this exercise, you will set the required DLL files as references in a .NET application.

1. Add a new project called **DotNetBank** to the Solution 'BankingClasses' in the Solution Explorer.
2. Add a new reference to the **DotNetBank** project by right-clicking on the **DotNetBank** project and then selecting the **Reference** command in the **Add** menu.
3. Check BankingClasses in the Projects tab.

4. In the Browse tab, click the Browse button and then add the following three references.

5. Navigate to C:\Projects\BankingClasses in the File Explorer and then open BankingClasses.exe.config in Visual Studio.
6. Copy the contents of `BankingClasses.exe.config` and then paste it into `app.config`.

```
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
    <add name="myDefault" providerName="JadeSoftware.Jobj.JobjConnectionString" connectionString="DataSource=Di\;jade\$\;1;In" />
  </connectionStrings>
  <job defaultConnection="myDefault">
    <installation directory="Di\;jade\$\;1\bin" />
  </job>
  <startup>
    <supportedRuntime version="4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
</configuration>
```

7. Ensure that the **Application** and **Build** settings are set to the same as those of the `BankingClasses` application; that is:
   - Target Framework is set to **.NET Framework 4.7.1** in the **Application** tab of the project properties.
   - Platform Target is set to **x64** in the **Build** tab of the project properties.

---

### C#, XAML, and the Windows Presentation Framework

When developing .NET applications, one typically uses the Windows Presentation Framework (WPF), which uses eXtensible Application Markup Language (XAML, pronounced "Zammel") to define the presentation of the application and the C# programming language to define the logic of the application.

XAML elements can also be created by manually typing in the required markup.
C# code is used to define the logic of the form; for example, the behavior when a button is clicked.

```csharp
using System.Windows;

namespace BankingApp
{
    /// <summary> Interaction logic for Example.xaml </summary>
    public partial class Example : Window
    {
        public Example()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

Exercise 5 – Creating a WPF Application

In this exercise, you will create a WPF application that will provide a .NET interface to the JADE banking system exposure. At first, you will simply display all customers of the bank in a list box.

1. Create a ListBox control in the center of the window using the Designer and then add Name="lbCustomers" to the ListBox element in the XAML.
2. Switch to Code View by pressing F7 and then add the following `using` statements.

```csharp
using System.Windows;
using BankingClasses;
using JadeSoftware.Joob;
using JadeSoftware.Joob.Client;
using JadeSoftware.Jade.DotNetInterop;
```

**Note** There may be superfluous `using` statements when the code is first opened. To remove them, select any `using` statement, press Alt+Enter, and then click **Remove Unnecessary Usings**.

3. Add the following code. This will create a reference called `bank`, which through the `JoobContext` object, refers to the `Bank` class defined in JADE.

```csharp
namespace DotNetBank
{
    public partial class MainWindow : Window
    {
        JoobContext context; // Declare a variable of type JoobContext (which is an inbuilt JADE class)
        Bank bank; // Declare a variable of type Bank (which is one of our classes from JADE)
        public InitializeComponent(); // Creates the window described in XAML
        context = new JoobContext(); // Instantiates the JoobContext, running the DotNetConnection application
        context = new JoobContext(); // that we made in JADE
        bank = context.FirstInstancedBank(); // Similar to Class.FirstInstance in Jade - gets the first Bank/
        // Remember, Bank is a singletion so there is always only one.
    }
}
```

**Note** The comments are for illustrative purposes and can be omitted.

4. To populate the `lbCustomers` list box with the customers of the bank, add the following to the `MainWindow` constructor method.

```csharp
lbCustomers.ItemsSource = bank.AllCustomers; // Listboxes display a collection, so we can just set
// the ItemsSource property to the allCustomers collection
lbCustomers.DisplayMemberPath = "LastName"; // Uses the LastName property of the Customer as the label
```

5. Press F5 to run the application. You should now see the last names of the bank’s customers displayed in the list box.