

JADE .NET Exposures

Version 2018

JADE Software Corporation Limited cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages, or loss of profits. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of JADE Software Corporation Limited.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Copyright © 2019 JADE Software Corporation Limited.

All rights reserved.

JADE is a trademark of JADE Software Corporation Limited. All trade names referenced are the service mark, trademark, or registered trademark of the respective manufacturer.

Contents

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

4 4

4

4

5

6

10

11

14

17

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.

1	Define Application	×
Application	Form Web Options	
Name	DotNetConnection	
Help File	Browse	
Version #		
Default <u>L</u> ocale	•	
Application Type	Non-GUI	
Web Application 1	ype	
JADE Forms	HTML Documents Web Services	
Icon	<u>C</u> hange Clea <u>r</u>	
Startup Form	T	
About Form	~	
Ini <u>t</u> ialize Method	Show Super Class Methods	
BankingModelSchen	a::initialize 🔹	
<u>F</u> inalize Method	Ŧ	
	OK Cancel <u>H</u> elp	

- a. Enter **DotNetConnection** in the Name text box.
- b. Select Non-GUI in the Application Type drop-down list box.
- c. 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.
- d. 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.

J	BankingViewSchema_Completed Exposure Browser	_	×
1	Web Services		

2. Name the exposure **BankingClasses** and then select **BankingModelSchema** option in the **Superschemas up to** drop-down list box.

J	C# Exposure Wizard : BankingClasses	□ ×
	Define Exposure	
	Exposure Name BankingClasses Current schema only Superschemas up to Select Top Most Schema BankingModelSchema_Completed NameSpace (Exposure Name used if empty) BankingClasses	
Step 1 of 6	< Back Next > Cancel	Help

- 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.

J	C# Exposure Wizard : BankingClasses		_ 🗆 ×
	Select Classes		
Jade Classes		Selected Classes for C# Bank BankingModelSchema CustomerByLastName	
Step 2 of 6	< <u>B</u> ack	<u>N</u> ext > Cancel	Help

Note Selecting only the classes that you intend to use provides a simpler interface to JADE. You can always add additional classes to the exposure later, if required.

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.

	C# Exposure Wizard : BankingClasses	
	Select Features for Classes	
Classes		<u>F</u> eatures
Jade Class	C# Class name	🖹 Bank
Bank	Bank	allChequeAccounts
BankingModelSchema	BankingModelSchema	✓ allCustomers
Customer	Customer	
CustomerByLastName	CustomerByLastName	myBankAcctSeqNum
		nextBankAcctNum
		nextCustomerNum
3 of 6	< Back	<u>N</u> ext > Cancel <u>H</u> elp

5. In the **BankingModelSchema** class, check the **myBank** feature.

		Frederic
<u>Classes</u> Jade Class	C# Class name	Eeatures
Bank	Bank	✓ myBank
BankingModelSchema	BankingModelSchema	initialize
Customer	Customer	
CustomerByLastName	CustomerByLastName	

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

<u>C</u> lasses		Eeatures
Jade Class	C# Class name	Customer
Bank	Bank	address allBankAccounts
BankingModelSchema	BankingModelSchema	I alibanitAccounts
Customer	Customer	✓ lastName
CustomerByLastName	CustomerByLastName	myBank
		✓ number
		setPropertiesOnCreate

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.

J	C# Exposure Wizard	: BankingClasses	□ ×
		Generate	
	Output Directory for C# Classes D:\Projects\BankingClasses\ Generate Sample .csproj File Generate Sample .config File Jade Database Path Jade Initialization File Path Sign-on Schema Name Sign-on Application Name	D:/jadeUni/dotnet/ D:/jadeUni/dotnet/jade.ini BankingViewSchema DotNetConnection Image: Sign-on to Jade as Multiuser	Br <u>o</u> wse Brows <u>e</u>
Step 6 of 6		< Back Generate Close	Help

e. Check the Sign-on to Jade as Multiuser check box.

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.

<mark> </mark> 🎝	🗖 🗧 🛛 course					— 🗆	×
File	Home Share	View					~
Pin to Quic access	k Copy Paste	🖌 Move to 🔻	➤ Delete ▾ ➡ Rename	New folder	Properties	Select all	'n
	Clipboard	Orga	nize	New	Open	Select	
$\leftarrow \rightarrow$	 This PC 	> DATA (D:)	> Projects >	course	✓ C Search c	ourse	Q
^	Name	^	Da	ate modified	Туре	Size	
	bin		31	/08/2018 1:55 PI	M File folder		
	obj		31	/08/2018 1:55 PI	M File folder		
	Properties		31	/08/2018 1:52 PI	M File folder		
	Bank.cs		31	/08/2018 1:52 PI	M Visual C# Sou	rce F 7	KB
	C# BankingClasses.	csproj	31	/08/2018 1:52 PI	M Visual C# Proj	ect 5	KB
- 64	BankingClasses.	exe.config	31	/08/2018 1:52 PI	M XML Configur	ratio 1	KB
	BankingModelSo	hema.cs	31	/08/2018 1:52 PI	M Visual C# Sou	rce F 7	KB
	Customer.cs		31	/08/2018 1:52 PI	M Visual C# Sou	rce F 11	KB
	CustomerByLast	Name.cs	31	/08/2018 1:52 PI	M Visual C# Sou	rce F 15	KB
~							
9 items	1 item selected 4.98	KB					:== =

2. Open BankingClasses.csproj in Microsoft Visual Studio 2017.



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

3. Press Alt+Enter to open the Properties menu.

nkingClasses 👳 🗙		
Application Build	<u>C</u> onfiguration: N/A	for <u>m</u> : N/A \lor
Build Events	Assembly <u>n</u> ame:	Defau <u>l</u> t namespace:
Debug	BankingClasses	BankingClasses
Resources	Target framework:	O <u>u</u> tput type:
Services	.NET Framework 4 🗸	Class Library 🗸
Settings Reference Paths Signing Code Analysis	.NET Framework 2.0 .NET Framework 3.0 .NET Framework 3.5 .NET Framework 3.5 Client Profile .NET Framework 4 .NET Framework 4 Client Profile .NET Framework 4.5.1 .NET Framework 4.5.2 .NET Framework 4.6.1 .NET Framework 4.6.1 .NET Framework 4.7.1	Assembly Information
	.NET Framework 4.7.2 Install other frameworks	V Browse
	Manifest: Embed manifest with default settings	×

Change the Target framework to .NET Framework 4.7.1.

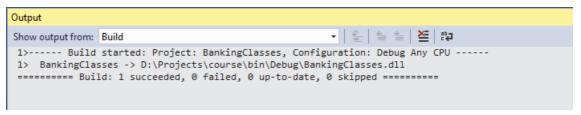
4. In the following message box that is then displayed, click **Yes**.

Target Fra	amework Change	×
?	Changing the Target Framework requires that the current project be closed and then reopened. Any unsaved changes within the project will be automatically saved.	
	Changing Target Framework may require manual modification of project files in order to build.	
	Are you sure you want to change the Target Framework for this project?	
	Yes No Help	
	ite inter	

Application Build*	Configuration: Active (Debug) V Platform: Active (Any CPU) V	
Build* Build Events Debug Resources Services Settings Reference Paths Signing Code Analysis	General Conditional compilation symbols: Define DEBUG constant Define TRACE constant Platform target: Any CPU Prefer 32-bit X86 Optimize code x84 Optimize code Errors and warnings Warning level: 4 Suppress warnings: Treat warnings as errors	
	None All Specific warnings:	

5. Click the **Build** tab, and then change the platform target from **Any CPU** to **x64**.

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



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.

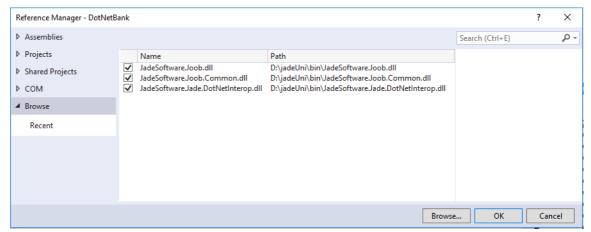
2. Add a new reference to the **DotNetBank** project by right-clicking on the **DotNetBank** project and then selecting the **Reference** command in the **Add** menu.

Solution Explorer (Ctrl-2) Image: Solution Explorer (Ctrl-2)	Solution Explorer		▼ ₽ ×				
Search Solution Explorer (Ctrl-:) > Image: Solution BankingClasses > Image: Solution Explorer >		- 4	· C "				
Solution 'BankingClasses' (2 pre-							
Debug Debug Solution Explorer Solution Explorer Solution Explorer Solution Explorer Cut Debug Initialize Interactive with Project Solution Explorer Solution Explorer Cut Ctrl+X Paste Ctrl+V Rename Unload Project Unload Project Open Folder in File Explorer Design in Blend	Solution 'BankingClasses' (2 prc ▲ ▲ C# BankingClasses ▶ Properties ▶ ■■ References ▶ C# BankingModelSchema.c: ▶ C# Customer.cs						
Image: Analyze Image		*	Build				
Analyze Publish Publish Scope to This Cope to This Researce Add Ad Ad Add Ad Add Ad Add Ad Ad Add Ad A			Rebuild				
Image Analyze Image Image Publish Scope to This Image Scope to This New Solution Explorer View Image New Solution Explorer View Image New Solution Explorer View Image Add Image Add Image New Solution Explorer View Image New Solution Explorer View Image Manage NuGet Packages Image Set as StartUp Project Image Set as StartUp Project Image New Folder Image Solution Explorer Image Source Control Image Cut Image Cut Image Source Control Image Paste Image Image							
Image: Solution Explorer View New Solution Explorer View Image: Presentation Explorer View Build Dependencies Image: Presentation Explorer View Build Dependencies Image: Nucker System Add Image: System Add Image: System Manage NuGet Packages Image: System Set as StartUp Project Debug Image: Source Control Image: Source Control Image: Source Control Image: Source Control Set as StartUp Project Image: Source Control Set as StartUp Project Image: Paste Ctrl+X Image: Rename Image: Rename Image: Paste Ctrl+V Rename Image: Paste Image: Paste Ctrl+V Image: Rename Image: Paste Image: Paste Ctrl+V Image: Paste Image: Paste Image: Paste Image: Paste Image: Paste Image: Paste Image: Paste Image	-	<i>.</i>		•			
 Microso Presenta Presenta Presenta Presenta Presenta Presenta System Add Add New Item Ctrl+Shift+A Existing Item Shift+Alt+A Existing Item		Ð					
Image: Presentation Protect Presentation Build Dependencies Image: System Add Image: NuGet Packages Image: System Manage NuGet Packages Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Set as StartUp Project Image: NuGet Packages Image: System Cut Ctrl+X Source Control Image: NuGet Packages Veb Reference Source Control Image: NuGet Package Veb Reference Image: Source Control Image: NuGet Package Veb Reference Image: Source Control Image: NuGet Package Veb Reference Image: Source Control Image: NuGet Package Image: NuGet P		_					
Build Dependencies Image NuGet Packages Image Nuget Pack							
Add Image NuGet Packages Image Nuget Packages <t< th=""><th></th><th></th><th colspan="2">Build Dependencies</th><th></th><th></th><th></th></t<>			Build Dependencies				
Image: System. Image	■■ System		Add	•	*ם	New Item	Ctrl+Shift+A
Set as StartUp Project Image: Set as StartUp Project <		Ě	Manage NuGet Packages		ta	Existing Item	Shift+Alt+A
Solution Explorer Team Debug Initialize Interactive with Project Source Control Source Control Cut Paste Ctrl+X Paste Remove Rename Unload Project Unload Project Open Folder in File Explorer Design in Blend Remove Del Cut Cut Cut Cut Cut Cut Cut Cut Del Rename Unload Project Cut Design in Blend	· · · · · · · · · · · · · · · · · · ·	ø	Set as StartUp Project	Jp Project		New Folder	
Initialize Interactive with Project Web Reference Source Control Service Reference Cut Ctrl+X Paste Ctrl+V Remove Del Rename Image:			Debug	•		Reference	
Source Control Service Reference Cut Ctrl+X Paste Ctrl+V Remove Del Rename Image: Imag	Solution Explorer Tean		Initialize Interactive with Project			Web Reference	
Ab Cut Cut+x Paste Ctrl+V Analyzer Remove Del Image: Cut+x Image: Cut+x Vindow Rename Image: Cut+x Unload Project Image: Cut+x Open Folder in File Explorer Image: Cut+x Design in Blend Image: Cut+x				•		Service Reference	
Paste Ctrl+V Analyzer Kemove Del Window Rename Page Unload Project User Control Open Folder in File Explorer Resource Dictionary Design in Blend Resource Dictionary		¥	Cut	Ctrl+X	t∌	Connected Service	
Remove Del Image: Window Rename Image: Page Unload Project Image: User Control Open Folder in File Explorer Image: Resource Dictionary Design in Blend Image: Class						Analyzer	
Rename Page Unload Project User Control Open Folder in File Explorer Resource Dictionary Design in Blend Class			Remove	Del		Window	
Opioad Project Implementation Open Folder in File Explorer Implementation Design in Blend Class	🗵 Rename		Rename		₽	Page	
Open Folder in File Explorer Resource Dictionary Design in Blend Class					2 7	User Control	
Design in Blend					Resource Dictionary		
					妆	Class	
		۶		Alt+Enter	-		

3. Check BankingClasses in the Projects tab.

Reference Manager - DotNetE	Bank			?	×
Assemblies			Search (Ctrl+E)		ρ.
Projects	Name	Path	Name:		
Solution	 BankingClasses 	D:\Projects\course\BankingClasses.csp	oroj BankingClasses		
Shared Projects					
▶ COM					
▷ Browse					
			Browse OK	Can	cel

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**.



- 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

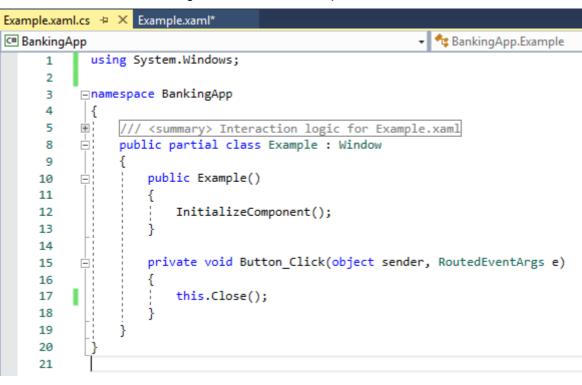
When developing .NET applications, one typically uses the Windows Presentation Framework (WPF), which uses eXtensible Application Markup Language (XAML, pronounced "Zammel") to define the presentation of the application and the C# programming language to define the logic of the application.

oolbo	х т म ×	Example.xaml.cs*	Example.xaml* 👳 🗙
arch	Toolbox 🔑 -	00000000	
Com	imon WPF Co 🔺	00000000	
- Ne	Pointer		
Ц	Border		ample
R	Button		
\checkmark	CheckBox		
	ComboBox		
=	DataGrid		
	Grid		
	lmage		
Α	Label		Button
E.	ListBox		
Θ	RadioButton		
	Rectangle		
旦	StackPanel		
- .	TabControl		
Τ	TextBlock	L	
abi	TextBox	00000000	

XAML can be generated in Visual Studio by using the Designer - a What-You-See-Is-What-You-Get (WYSIWYG) editor.

XAML elements can also be created by manually typing in the required markup.

🕞 Design	↑↓ 💿 XAML
🖸 Button	•
1	⊡ <window <="" td="" x:class="BankingApp.Example"></window>
2	<pre>xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"</pre>
3	<pre>xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"</pre>
4	<pre>xmlns:d="http://schemas.microsoft.com/expression/blend/2008"</pre>
5	<pre>xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"</pre>
6	<pre>xmlns:local="clr-namespace:BankingApp"</pre>
7	<pre>mc:Ignorable="d"</pre>
8	Title="Example" Height="300" Width="200">
9	<pre>Grid></pre>
10	<button click="Button_Click" content="Button" height="30" width="50"></button>
11	
12	
13	
14	



C# code is used to define the logic of the form; for example, the behavior when a button is clicked.

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.

🕞 Design	↑↓ I XAML
🖸 ListBox	
1	
2	<pre>xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"</pre>
3	<pre>xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"</pre>
4	<pre>xmlns:d="http://schemas.microsoft.com/expression/blend/2008"</pre>
5	<pre>xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/200</pre>
6	<pre>xmlns:local="clr-namespace:DotNetBank"</pre>
7	<pre>mc:Ignorable="d"</pre>
8	Title="MainWindow" Height="450" Width="800">
9	□ <grid></grid>
10	<listbox height="200" name="lbCustomers" width="300"></listbox>
11	
12	

2. Switch to Code View by pressing F7 and then add the following **using** statements.

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.



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

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

22			1			
23		÷.			<pre>lbCustomers.ItemsSource = bank.AllCustomers;</pre>	<pre>// Listboxes display a collection, so we can just set</pre>
24				1		<pre>// the ItemsSource property to the allCustomers collection</pre>
25			- i -			
26					lbCustomers.DisplayMemberPath = "LastName";	<pre>// Uses the LastName property of the Customer as the label</pre>
27						// in the ListBox.
28			1	}		
29			}			
30	- [}				
31						

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