
Contents

Introducing the Eclipse-based LotusScript Editor	1
Exercise 1: Exploring the Eclipse-based LotusScript editor.	3
Exercise 2: Using code and comment templates.	5
Exercise 3: Using tools to assist you when writing scripts	7
Exercise 4: Working with and writing code	9
Exercise 5: Supporting the Eclipse-based LotusScript editor	12
Appendix A. Answer keys	13

Answer key for Exercise 1: Exploring the Eclipse-based LotusScript editor	14
Answer key for Exercise 2: Using code and comment templates.	15
Answer key for Exercise 4: Working with and writing code	16

Appendix B. Checklist for critical objectives	17
--	-----------

Introducing the Eclipse-based LotusScript Editor

In this module, you will learn to use the Eclipse-based LotusScript® Editor that was introduced in IBM® Lotus® Domino® Designer 8.5.1.

Time required

It should take approximately two hours to complete this module.

Prerequisites and system requirements

To complete this module, you will need the following:

- Knowledge of IBM Lotus Domino Designer 8.5
- An All Clients installation of IBM Lotus Notes® 8.5.1, including the Lotus Notes client, IBM Lotus Domino Administrator, and Lotus Domino Designer
- An 8.5.1 IBM Lotus Domino server

Learning objectives

After completing this module, you should be able to:

- Launch the Eclipse-based LotusScript editor.
- Disable the Eclipse-based LotusScript editor.
- Configure the Eclipse-based LotusScript editor to display line numbers for code.
- **Critical Objective:** Create a LotusScript agent using the Eclipse-based LotusScript editor.
- Use the continuous error reporting feature to locate and resolve an error in a line of code.
- **Critical Objective:** Use a code template to automatically insert code into a design element.
- Use a comment template to automatically insert a comment into a design element.
- Use a variable in a comment template.
- Use content assist to complete a statement.
- Use hover help to learn more about a particular element.
- Use a hyperlink to access an element from within another element.
- **Critical Objective:** Save a script with errors.
- Describe how the find and replace function works in the Eclipse-based editor.
- Describe the options for importing and exporting LotusScript in IBM Lotus Domino Designer 8.5.1.

Critical Objectives

Some of the learning objectives in this module have been labeled as *Critical Objectives*. This term identifies learning objectives that were deemed critical to the success of students in their understanding of the topic. You can self-validate your critical objectives using the answer keys which are available in Appendix A. Appendix B contains a checklist of all of the critical objectives in this module along with the steps to validate your success.

Timing

The approximate timing for each exercise in this module is listed in the following table.

Table 1. Table of exercises in this module

Exercise	Approximate timing in minutes
Exercise 1: Exploring the Eclipse-based LotusScript editor	30
Exercise 2: Using code and comment templates	30
Exercise 3: Using tools to assist you when writing scripts	15
Exercise 4: Working with and writing code	30
Exercise 5: Supporting the Eclipse-based LotusScript editor	15
	Total: 2 hours

Exercise 1: Exploring the Eclipse-based LotusScript editor

In this exercise, you will work with the Eclipse-based LotusScript editor.

Time needed

It will take approximately 30 minutes to complete this exercise.

Objectives

After completing this exercise, you should be able to:

- Launch the Eclipse-based LotusScript editor.
- Disable the Eclipse-based LotusScript editor.
- Configure the Eclipse-based LotusScript editor to display line numbers for code.
- **Critical Objective:** Create a LotusScript agent using the Eclipse-based LotusScript editor.
- Use the continuous error reporting feature to locate and resolve an error in a line of code.

Working in the Eclipse-based LotusScript editor

Complete the following steps to begin exploring the Eclipse-based LotusScript editor.

1. Read the topic *Comparing the Eclipse-based and classic LotusScript editors* in the Lotus Domino Designer 8.5.1 Help.

Note: This help topic can be found under **Lotus Domino Designer XPages and Eclipse User Guide → Eclipse-based language interfaces → Eclipse-based LotusScript editor**.

2. Open IBM Lotus Domino Designer 8.5.1. Open the mail database for one of the users on your server in Lotus Domino Designer.
3. Open a script library in the mail database, such as CoreEmailClasses. The script library should launch in the Eclipse-based LotusScript editor.

Tip: The script libraries can be found under the **Code** category.

4. Spend 2 minutes exploring the updated editor. In particular, note the following:
 - Unique icons are used to indicate functions, sub-functions, and classes.
 - Different colors are used to flag the elements of a script, such as Identifiers, Keywords, and Comments.
 - You're able to expand the declarations so that they display in the Objects tab.

Configuring preferences

Complete the following steps to configure the preferences for the Eclipse-based LotusScript editor.


1. If you prefer to use the classic LotusScript editor, you can do so while still working in Lotus Domino Designer 8.5.1. To revert to the classic editor, open the preferences and click **Domino Designer → LotusScript Editor**. Clear the **Use Eclipse-based LotusScript editor** preference and then click **OK**.
2. Close and reopen the CoreEmailClasses script library. It should now open in the classic LotusScript editor.
3. Return to the preferences and enable the preference to use the Eclipse-based LotusScript editor.
4. Displaying line numbers next to each line of code is often useful when debugging or collaborating on code. Because the Eclipse-based LotusScript editor is based on an Eclipse Text Editor, the options available for text editors are also available in the LotusScript Editor. Enable line numbers by opening the preferences and clicking **General → Editors → Text Editors → Show line numbers**. Click **OK**.
5. Look at the CoreEmailClasses script library, and you should now see a line number next to each line of code.

6. The colors that are used to flag the elements of a script are also configurable. Open the preferences and click **Domino Designer** → **LotusScript Editor** → **Fonts and Colors**.
7. Change the colors for the various text types using this page. Use the preview window to see the changes. After you have tested several options, click the **Restore Defaults** button so that you're using the default colors.

Creating LotusScript agents

1. The Eclipse-based LotusScript editor can also be used to create LotusScript agents. Using the Eclipse-based LotusScript editor, create a LotusScript agent to delete the calendar profile document in the mail database that you have been working with.

Tip: The code is available in technote 1088892: *How to delete Profile documents manually or using LotusScript*.

2. When pasting the code into the agent, the indentation may not be in the correct format. Although this format won't prevent the agent from executing successfully, it's a good practice to use the Eclipse-based LotusScript editor formatting conventions. To automatically format your code, select all of the lines of code that you want to correct and click **Source** → **Correct Indentation** or CTRL-I.
3. Error reporting is slightly different in the updated editor. To simulate an error, remove the quotes that surround the name of the profile document in your code. Note the various places where the error icon  displays to alert you that there is a problem with the code. This feature is referred to as *continuous error reporting*. Also note that you can hover over the error icon to display the error message.
4. Restore the quotes to resolve the problem with the code and save your agent.

Exercise 2: Using code and comment templates

In this exercise you will learn to use templates to automatically insert code or comments into your LotusScript code. These are two features introduced with the Eclipse-based LotusScript editor in IBM Lotus Domino Designer 8.5.1.

Time needed

It will take approximately 30 minutes to complete this exercise.

Objectives

After completing this exercise, you should be able to:

- **Critical Objective:** Use a code template to automatically insert code into a design element.
- Use a comment template to automatically insert a comment into a design element.
- Use a variable in a comment template.

Working with comment templates

Complete the following steps to learn how to use comment templates.

1. Read the topics *Controlling comment generation* and *Code and comment template variables* in the Lotus Domino Designer 8.5.1 Help.

Note: These help topics can be found under **Lotus Domino Designer XPages and Eclipse User Guide** → **Eclipse-based language interfaces** → **Eclipse-based LotusScript editor** → **Working with code and comment templates**.

2. Open the **Comment Templates** section of the Lotus Domino Designer preferences by clicking **File** → **Preferences** → **Domino Designer** → **LotusScript Editor** → **Comment Templates**
3. Ensure that the option to **Automatically include comments in new code elements** is checked. This option should be enabled by default.
4. Explore the various code elements for which you can configure Lotus Domino Designer to automatically include comments and then select **Design Element**.
5. Add the following comment to the **Comment template** using variables: Please do not modify the code in the <database name> application. Contact <author's name> for change requests to this code.

Tip: Use the **Insert Variable** button for a list of available variables. Double-click a variable in the list to add it to the template.

6. Check the **Preview** box to display the code as it will appear when added to a design element. Click **Apply** and then **OK** to save your changes.
7. Create a new LotusScript agent with a target of **None** in the mail database that you worked with in the previous exercise. You should see that the comments that you configured are inserted automatically.
8. Open an existing agent in the application that is not the agent that you just modified. Notice that the comments are not automatically added to existing design elements. You can manually add the comments from the template by clicking **Source** → **Apply comment template**.

Working with code templates

Complete the following steps to work with code templates.

1. Read the topic *Creating and using code templates* in the Lotus Domino Designer 8.5.1 Help.

Note: This help topic can be found under **Lotus Domino Designer XPages and Eclipse User Guide → Eclipse-based language interfaces → Eclipse-based LotusScript editor → Working with code and comment templates.**

2. Consider a scenario where an application developer frequently writes code to perform operations on the current database. To save time, the application developer could create a code template to establish a database object for the current database when creating a new sub. Configure a code template for subs with the following code:

```
Dim session As New NotesSession
Dim db As NotesDatabase
Set db = session.CurrentDatabase
```

Tip: To access code templates, click **File → Preferences → Domino Designer → LotusScript Editor → Code Templates.**

3. Create a new agent in the mail database that you have been working with. Create a new sub in the agent named test: Sub test(). Notice that when you press the **Enter** key after typing the sub definition, the lines from the code template are automatically added. Demonstrate this behavior to your facilitator.

Tip: One way to create a new sub is to:

- a. Click Declarations.
 - b. In a new line, type Sub test().
 - c. Press enter.
4. Add this code below the lines that were added automatically from the code template:

```
Forall m In db.Managers
Messagebox(m)
End Forall
```
 5. Click the Initialize() sub. Notice that the code was not automatically added to this sub and that in the **Source** menu, your only option is to apply the comment template, not the code template.
 6. Call Sub test() from Sub Initialize() using the code Call test so that you have a functioning agent. Save your changes.

Exercise 3: Using tools to assist you when writing scripts

In this exercise you will learn to use three tools to help you while you're writing code in the IBM Lotus Domino Designer 8.5.1 Eclipse-based LotusScript editor: content assist, hover help, and hyperlinks.

Time needed

It will take approximately 15 minutes to complete this exercise.

Objectives

After completing this exercise, you should be able to:

- Use content assist to complete a statement.
- Use hover help to learn more about a particular element.
- Use a hyperlink to access an element from within another element.

Using content assist

Complete the following steps to learn how to use content assist.

1. Read the topic *Using content assist* in the Lotus Domino Designer 8.5.1 Help.

Note: These help topics can be found under **Lotus Domino Designer XPages and Eclipse User Guide** → **Eclipse-based language interfaces** → **Eclipse-based LotusScript editor** → **Using help in the user interface**.

2. Open the agent that you worked with in the previous exercise. Create a new sub named test2: Sub Test2(). Press enter and confirm that the code from the code template is added to the sub.
3. Test content assist by entering the following code: Messagebox db. (including the period). Notice that a menu displays with the options for completing this line of code. Click **Filename As String** once. Notice that this action populates the window with a description of this option.

Tip: Type the code in manually to display the menu. If you use copy and paste the menu will not display.

4. Double-click **Filename As String** to add the code to your script.
5. Type a comma, and then type db. (including the period). Click the script editor to cancel out of content assist. Type a t after the period to continue writing your script. Content assist is not invoked because you previously cancelled it for this piece of code. Manually invoke content assist by entering CTRL-space and select **Title As String** to add the code to your script. Notice that because you had already typed the "t" after the period, content assist automatically filtered the suggested elements to those that begin with a "t".
6. Save your changes.

Working with hover help

Complete the following steps to learn about using hover help.

1. Read the topic *Using hover help* in the Lotus Domino Designer 8.5.1 Help.

Note: These help topics can be found under **Lotus Domino Designer XPages and Eclipse User Guide** → **Eclipse-based language interfaces** → **Eclipse-based LotusScript editor** → **Using help in the user interface**.

2. Open the CoreEmailClasses script library in the mail database that you have been working with.
3. Hover your mouse cursor over the elements in this script library, such as keywords, functions, and variables to see the hover help that displays for each type of element.
4. Open the agent that you worked with in the previous part. Hover over Sub Test2(). Note that the hover help displays the name of the agent that contains the sub.

5. Now enter a description for this sub in the comments section above Sub Test2(), such as This sub displays the filename and title of the current application in a messagebox.
6. Save your changes and then hover over Test2(). Notice that now the comments that you entered display in the hover help for that sub.
7. The delay before the editor displays the content assist or hover help is configurable in the preferences. Explore the options that are available by clicking **File** → **Preferences** → **Domino Designer** → **LotusScript Editor**. The configuration options are available in the **Code Assistance** section of this page.

Using hyperlinks

Complete the following steps to learn how to use hyperlinks when writing LotusScript.

1. Read the topic *Using hyperlinks* in the Lotus Domino Designer 8.5.1 Help.

Note: These help topics can be found under **Lotus Domino Designer XPages and Eclipse User Guide** → **Eclipse-based language interfaces** → **Eclipse-based LotusScript editor** → **Using help in the user interface**.

2. Open the CoreEmailClasses script library in the mail database that you've been working with.
3. Click on the QuerySaveExtension in the Objects tab so that it displays in the editor.
4. Locate the line of code that calls the FaxToProc sub. Notice that this code displays as any other text. If you hover over the sub name, however, the hover help displays.
5. Many times, it is helpful to quickly refer back to the sub so that you can take a closer look at what the sub does. Either CTRL-click on the FaxToProc sub or click on the FaxToProc sub and press F3. You should see that this action takes you directly to the sub.

Exercise 4: Working with and writing code

In this exercise you will learn how several functions that are often used when working with and writing code function in the Eclipse-based LotusScript editor.

Time needed

It will take approximately 30 minutes to complete this exercise.

Objectives

After completing this exercise, you should be able to:

- **Critical Objective:** Save a script with errors.
- Describe how the find and replace function works in the Eclipse-based editor.
- Describe the options for importing and exporting LotusScript in IBM Lotus Domino Designer 8.5.1.

Saving a script with errors

Complete the following steps to test saving a script with errors.

1. Read the topic *Allowing script documents to be saved with errors* in the Lotus Domino Designer 8.5.1 Help.

Note: This help topic can be found under **Lotus Domino Designer XPages and Eclipse User Guide** → **Eclipse-based language interfaces** → **Eclipse-based LotusScript editor**.

2. Open the **LotusScript Editor** preferences and confirm that the **Saving With Errors** options are set to their default values:
 - a. **Prompt me on save if there are LotusScript errors** should be selected.
 - b. Clear the check for **Remove existing object code when saving with errors**.

Tip: The LotusScript Editor preferences can be opened by clicking **File** → **Preferences** → **Domino Designer** → **LotusScript Editor**.

3. Open the mail database that you have been using for testing in the IBM Lotus Notes Client. Run the agent that you created for testing in a previous exercise. You should see that the name of each Manager in the ACL of the database displays in a message box.
4. Now open the agent in the database using Lotus Domino Designer.
5. Cause an error in Sub test() by adding a semicolon to the line Messagebox(m). You should immediately see the red X icon from the continuous error reporting feature alerting you that there is a problem with the code.
6. Save your agent with the error in the code. You will receive a prompt that says Script contains errors. Would you like to save anyway? because your preferences are configured to prompt you on save if there are LotusScript errors. Click **Yes** to save the agent with errors.
7. Run the agent using the Lotus Notes Client. Notice that the agent still executes successfully even though the script was saved with an error. This behavior occurs because Lotus Domino Designer is currently not configured to remove the existing object code when saving with errors. This configuration preserves the last version of the script that was saved without errors to use at runtime.
8. Now configure Lotus Domino Designer to discard the existing code when saving with errors by checking this option in the LotusScript preferences.

Tip: Refer to step 2 for help locating these preferences.

9. Add another semicolon to the line Messagebox(m) to generate another error and then save the agent. Click **Yes** when you are asked whether you want to save the script with errors.
10. Run the agent using the Lotus Notes Client. You will see that the agent still runs successfully using the version of the script that was saved without errors even though Lotus Domino Designer is not

currently configured to preserve the successfully compiled script. This behavior was reported to IBM Lotus Development in SPR JPIK7VJR7W and documented in technote 1403461.

Remember: At this point your Lotus Domino Designer preferences are configured to **Prompt me on save if there are LotusScript errors** and **Remove existing object code when saving with errors**.

11. Create a new agent that contains the good script. Save your changes and then run the agent from the Lotus Notes Client.

Note: Refer to the *Working with code templates* section of Exercise 2 for instructions on creating the agent.

12. Now cause an error in the code and save the script with errors. Attempt to run the agent from the Lotus Notes Client. The agent should not run since Lotus Domino Designer is not configured to preserve the last successfully compiled script and was not configured to do so during a previous save of the script.
13. As with previous versions, it is not possible to save a script with errors using the classic LotusScript editor. Scripts saved with errors using the Eclipse-based LotusScript editor will display the last successfully compiled script if one is available when viewed from classic LotusScript editor. If a successfully compiled script is not available, the script containing errors will not be displayed when the design element is accessed using the classic LotusScript editor. Take five minutes to test and observe this behavior.

Using find and replace

Complete the following steps to learn about some differences between using find and replace in the classic and Eclipse-based LotusScript editors.

1. When you use the find and replace feature to perform a search in the classic LotusScript editor, you're able to choose whether you want the scope to be limited to all objects, the current object, or only the current section. In the Eclipse-based LotusScript editor, the find and replace feature does not have this option. Use the Eclipse-based LotusScript editor to open the CoreEmailClasses script library in the mail database that you've been working with to test this functionality.
2. In the objects tab, select the Initialize sub.
3. In the editor, launch a find and replace by pressing CTRL-F or by clicking **Edit** → **Find/Replace**.
4. Search for FaxGetDomain. Notice that Sting Not Found is returned even though the function exists. This is because only the section selected in the objects tab is searched, which is Initialize in this example.
5. If you need to search the entire script library, select the top node in the objects tab, which displays all elements in the library, and perform the search.

Importing and exporting code

Complete the following steps to learn how to import and export code using the Eclipse-based editor.

1. Read the section *Importing and exporting code* in the topic *Comparing the Eclipse-based and classic LotusScript editors* in the Lotus Domino Designer 8.5.1 Help.

Note: This help topic can be found under **Lotus Domino Designer XPages and Eclipse User Guide** → **Eclipse-based language interfaces** → **Eclipse-based LotusScript editor**.

2. In the CoreEmailClasses script library, click in the editor pane. Examine the options available in the **File** menu. Note that neither **Import** nor **Export** is available.
3. Select the entire library by clicking **Edit** → **Select All**. Right-click on the selected text, and note that you are able to copy the code. This method is one way to move code between design elements as needed without using the import or export options.
4. In the LotusScript preferences, clear the option to **Use Eclipse-based LotusScript editor**. Click **Apply** and then **OK**.

5. Close and reopen the CoreEmailClasses library. Note that now both the **Import** and **Export** options are available in the **File** menu. Switching to the classic LotusScript editor is another option if a customer needs to import or export LotusScript code.

Exercise 5: Supporting the Eclipse-based LotusScript editor

In this exercise, you will learn more about supporting the Eclipse-based LotusScript editor. You will learn about some limitations and differences in behavior when using the Eclipse-based LotusScript editor versus the classic LotusScript editor.

Time needed

It will take approximately 15 minutes to complete this exercise.


Objectives

After completing this exercise, you should be able to:

- List several limitations when working with the Eclipse-based LotusScript editor.
- List several differences in behavior when working with the Eclipse-based LotusScript editor.

Testing limitations and differences in behavior with the Eclipse-based LotusScript editor

Complete the following steps to test several functions that are either not possible or behave differently when using the Eclipse-based LotusScript editor.

1. In the classic LotusScript editor, an application developer can change the type of an agent after it is created. For example, an agent that is created initially as a formula agent can later be changed to a LotusScript agent. This action is not possible in the Eclipse-based LotusScript editor. Using the Eclipse-based LotusScript editor, open one of the agents that you used for testing in previous exercises and observe that it is not possible to change the type of agent.
2. When working with scripts, an application developer may need to recompile all LotusScript in the application. In the classic LotusScript editor, this operation was performed by clicking **Tools** → **Recompile All LotusScript**. In the Eclipse-based LotusScript editor, this function is also available, as well as another menu option that performs the same task: **Project** → **Clean**. Locate both of these options in Lotus Domino Designer.
3. One feature that is not available in the classic LotusScript editor that application developers will likely find quite useful in the Eclipse-based LotusScript editor is the ability to maximize the script editor so that it fills the entire width of the Lotus Domino Designer window. Test this functionality by clicking on the **Maximize** icon: .
4. You may be wondering whether scripts saved with the Eclipse-based editor can be opened with the classic LotusScript editor. Any script saved with the Eclipse-based LotusScript editor should open using the classic editor with no problems. Test this functionality by configuring Lotus Domino Designer to use the classic LotusScript editor, and then open the agent that you created using the Eclipse-based LotusScript editor.

Remember: To use the classic LotusScript editor, click **File** → **Preferences** → **Domino Designer** → **LotusScript Editor** and clear the check for **Use Eclipse-based LotusScript editor** .

Important: As with previous versions of Lotus Domino Designer, modifying or running applications using an earlier version of Lotus Notes, Lotus Domino Designer, or the Lotus Domino server than the version that was used to originally create the application is not recommended or supported.

Appendix A. Answer keys

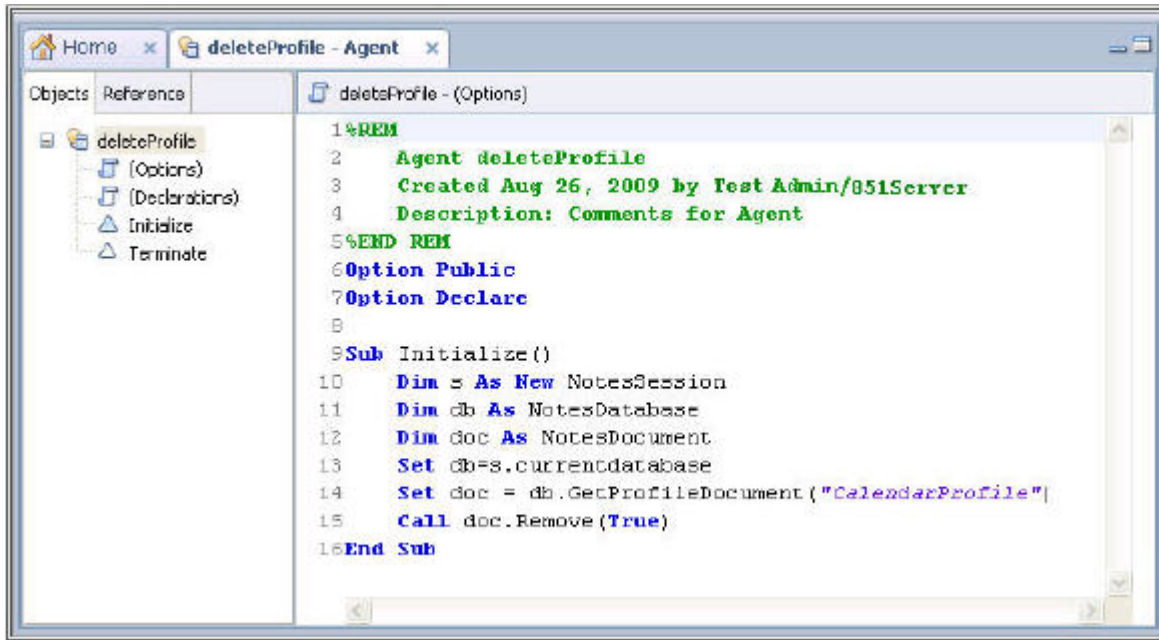
The following answer keys are available in this section:

1. Answer key for Exercise 1: Exploring the Eclipse-based LotusScript editor
2. Answer key for Exercise 2: Using code and comment templates
3. Answer key for Exercise 4: Working with and writing code

Exercises not listed here do not require an answer key.

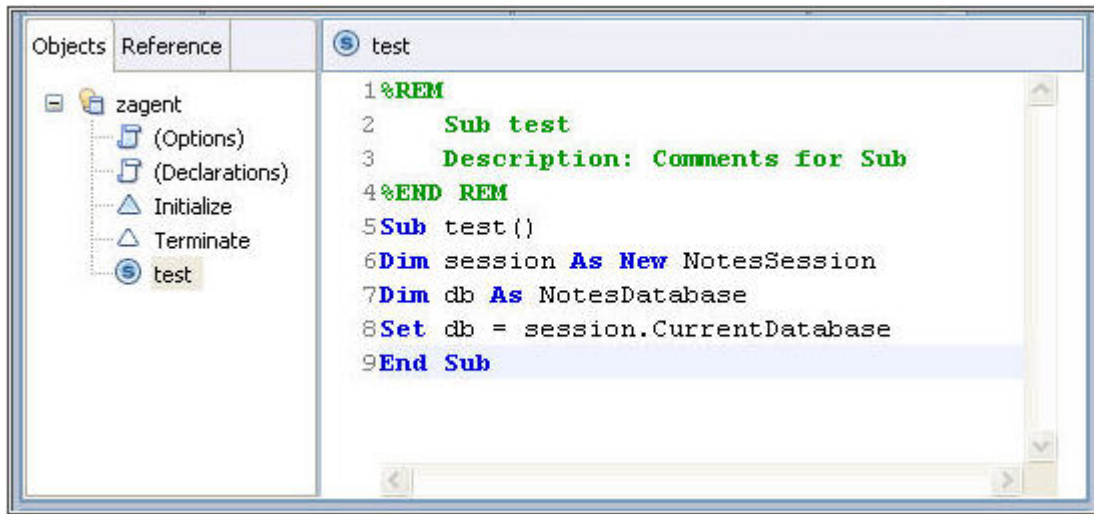
Answer key for Exercise 1: Exploring the Eclipse-based LotusScript editor

Your agent should look like the following screen capture:



Answer key for Exercise 2: Using code and comment templates

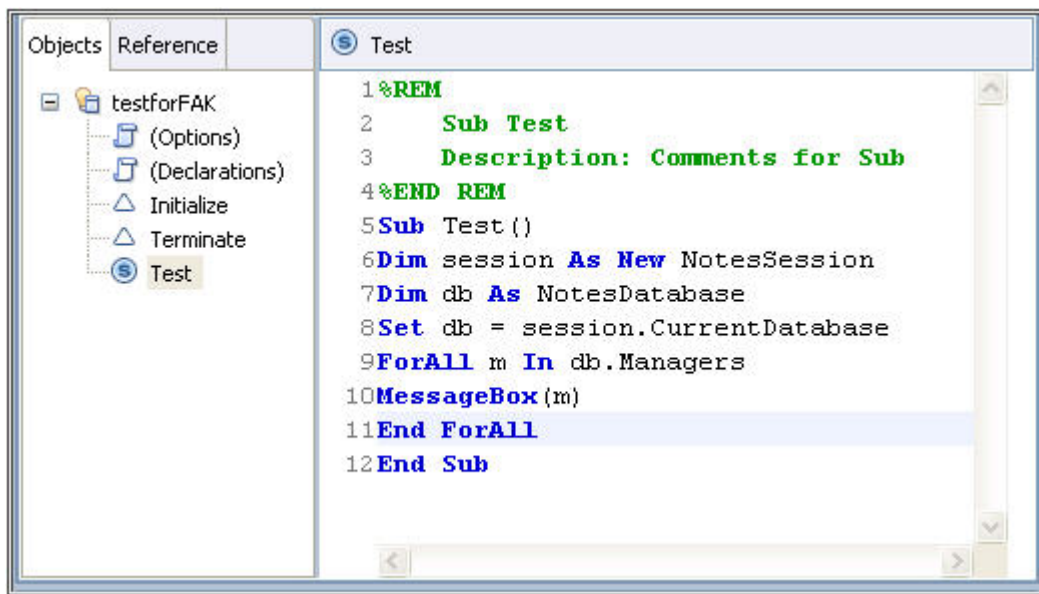
Your subs should look like the following screen capture after step 3 where you have created the code template and added code from that template to a new sub:



The screenshot shows the Visual Studio IDE with a project named 'zagent'. The 'Objects' pane on the left shows a tree view with 'test' selected. The main editor window displays the following code template for a sub named 'test':

```
1 %REM
2   Sub test
3   Description: Comments for Sub
4 %END REM
5 Sub test ()
6 Dim session As New NotesSession
7 Dim db As NotesDatabase
8 Set db = session.CurrentDatabase
9 End Sub
```

Your subs should look like the following screen capture after step 5 where you have added the additional code after the code that was added from the code template:

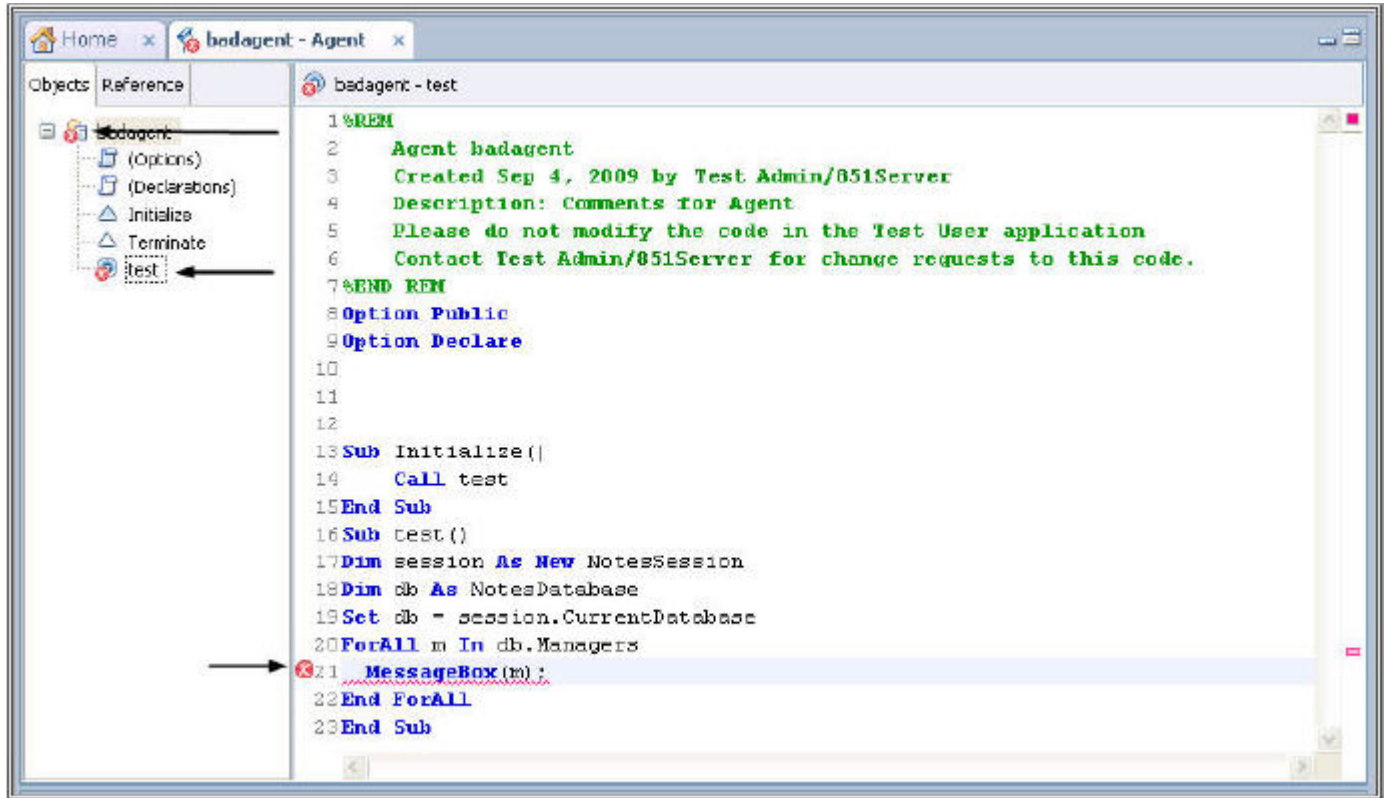


The screenshot shows the Visual Studio IDE with a project named 'testforFAK'. The 'Objects' pane on the left shows a tree view with 'Test' selected. The main editor window displays the following code for a sub named 'Test':

```
1 %REM
2   Sub Test
3   Description: Comments for Sub
4 %END REM
5 Sub Test ()
6 Dim session As New NotesSession
7 Dim db As NotesDatabase
8 Set db = session.CurrentDatabase
9 ForAll m In db.Managers
10  MessageBox (m)
11 End ForAll
12 End Sub
```

Answer key for Exercise 4: Working with and writing code

Your agent should look like the following screen capture when accessed using IBM Lotus Domino Designer:



Appendix B. Checklist for critical objectives

Table 2. Critical objectives for the *Introducing the Eclipse-based LotusScript Editor* module

Critical objectives	Ways to measure successful completion
<p>“Exercise 1: Exploring the Eclipse-based LotusScript editor” on page 3:</p> <ul style="list-style-type: none">• Create a LotusScript agent using the Eclipse-based LotusScript editor.	<p>Confirm that you successfully created the agent using the Eclipse-based LotusScript editor. One way to confirm this success is to compare your agent to the screen capture in the Answer key for Exercise 1: Exploring the Eclipse-based LotusScript editor.</p>
<p>“Exercise 2: Using code and comment templates” on page 5</p> <ul style="list-style-type: none">• Use a code template to automatically insert code into a design element.	<p>Confirm that you successfully created the code template and added code using that template to a new sub in the Eclipse-based LotusScript editor. One way to confirm this success is to compare your agent to the screen capture in the Answer key for Exercise 2: Using code and comment templates.</p>
<p>“Exercise 4: Working with and writing code” on page 9</p> <ul style="list-style-type: none">• Save a script with errors.	<p>Confirm that you successfully saved the script with errors. One way to confirm this is to look at the your application using IBM Lotus Domino Designer and check for an error icon next to the agent containing the script saved with an error. A screen capture is available in the Answer key for Exercise 4: Working with and writing code.</p>



Printed in USA