Using Code Templates in DDE

by Julian Robichaux, panagenda
originally published on socialbizug.org, July 2013

One of the “freebies” that came with integrating Domino Designer with the Eclipse platform (DDE) in Lotus Notes 8.5 is the concept of code templates. I mentioned these in my *Beginner Tips for Domino Designer in Eclipse* article for the Clippings newsletter way back in February of 2011, but I didn’t go into too much detail on how they can be used. Let’s take a look at some of the details now.

Templates for New Code and Comment Structures
There are actually two different kinds of code templates in Eclipse. The first is the template used for new comment blocks and code structures (classes, methods, properties, etc.). You have probably noticed that whenever you create a new LotusScript agent in DDE, it automatically looks like this:

```
%REM
	Agent Another Amazing LotusScript Agent
	Created Jul 30, 2013 by Your Name Here!
	Description: Comments for Agent
%END REM
Option Public
Option Declare
```

That comment block gets added because it has been defined as a comment template for “Design Element” in the DDE preferences. Similarly, if you type “class MyClass” and press Enter, the code will be auto-formatted as:

```
%REM
	Class MyClass
	Description: Comments for Class
%END REM
Class MyClass

End Class
```

because of the comment template for “Class”.

Where You Can Use These Templates
This type of template is available for LotusScript, JavaScript, and Java code. If you open the Notes Preferences dialog from DDE (File — Preferences) you will see the template definitions at:

- Domino Designer - LotusScript Editor - Code Templates
- Domino Designer - LotusScript Editor - Comment Templates
- Java - Code Style - Code Templates (there are separate entries for comments and code)
- JavaScript - Code Style - Code Templates (there are separate entries for comments and code)

Ideas for Customization
If you use LSDoc, you can very easily change the LotusScript comment templates to generate LSDoc-style syntax. See my *Use LotusScript.Doc to Document Your Code* article in the April 2011 Clippings newsletter for more information on using the excellent LSDoc tool.

Other ideas for comment template modifications include:

- Version number
- Modified date placeholder
- Code reviewer
- Copyright statements
- Open source license

The only thing I use this type of code template for is to automatically add logging in LotusScript (I use the other type of code templates for Java and JavaScript, discussed next). I have my LotusScript code templates set up like this:

<u>Design Element</u>
```
Use "OpenLogFunctions"
```

<u>Sub</u>
```
	On Error GoTo processError


	Exit Sub

processError:
	Call LogError()
	Exit Sub
```

<u>Function</u>
```
	On Error GoTo processError


	Exit Function
```

```
processError:
        Call LogError()
        Exit Function
```

This way all my new agents, script libraries, functions and subs automatically have error logging. Your logging library and coding style may vary, of course.

NOTE: you must make sure that the “Automatically include template in new code elements” option is checked in the Domino Designer - LotusScript Editor - Code Templates dialog. It is unchecked by default. If it’s not checked, your LotusScript code templates will not be used. Alternatively, you can uncheck the box for the Comment Templates if you really hate the auto-generated comments.

Templates for Content Assist/Code Completion

The second type of code template — and this type is what almost all Eclipse developers are talking about when they refer to “code templates” — is like a shortcut for frequently used blocks of code. It’s very similar to text expansion capabilities in some word processing programs or smartphones, where you type a short word or abbreviation and it is automatically substituted with text you commonly write, like a signature block or your address or a short text message reply. But with Eclipse there are also some very nice contextual options that make it even more effective.

Let’s start with a simple example. Open DDE and do the following:

1. Create a new Java agent
2. Place the cursor in the code editor just below the “// (Your code goes here)” comment
3. Type “sysout” (without the quotes)
4. Without moving the cursor (it should be directly after the “t” in “sysout” with no spaces) press the Control and Space keys at the same time
5. The word “sysout” will be replaced with “System.out.println();”

I know that some of you have already started saying nasty things in your head about how you should NEVER use System.out.println() in your code. Calm down, it’s just an example. Try it and see what it does, and then delete the offending line with malice if it bothers you.

Control+Space is the magic key combination here. Once you get used to using Control+Space in DDE, you’ll find yourself doing it in other programs too (like email) and wondering why it doesn’t work.

Where You Can Use These Templates
This type of template is available for JavaScript script libraries, Java agents and script libraries, HTML and XML File Resources, and CSS files. If you open the Notes Preferences dialog from DDE (File — Preferences) you will see the template definitions at:

- Java - Editor - Templates
- JavaScript - Editor - Templates
- Web - CSS Files - Editor - Templates
- Web - HTML Files - Editor - Templates
- XML - XML Files - Editor - Templates

You will notice that there is NOT an option for LotusScript, at least as of Notes 9.0.

Also, these templates do not work everywhere that you can use Java, JavaScript, HTML, etc. Whether or not they work depends on which Eclipse editor you are using. For example, the XML templates sorta-kinda work with the XML in the source view of XPages, but XPages uses a "special" XML editor for XPages and I've never had very consistent results with code templates in that editor. The JavaScript templates work with JavaScript script libraries but not with many of the other places you could enter JavaScript code, like SSJS in an XPage.

Useful Default Template Shortcuts
Here are some very useful template shortcuts that are already in the DDE preferences by default. Again, just type the shortcut word and press Control+Space. If there are multiple options for a shortcut (like "for" in Java) you can use the up and down arrow to navigate to the one you want, then press Enter.

Also, most of these have placeholders for variables within the code that is created. Use the tab key to go from variable to variable, filling them in as you go.

<u>Java, JavaScript: for</u>
Creates the structure of a "for" loop in one of several different ways: looping through an array, a collection, or an Iterable value. If there is already an array or collection or Iterable near the loop, that value will be pre-filled in the statement by default.

<u>Java, JavaScript: new</u>
A quick way to generate a new instance of any object.

<u>Java: static_final</u>
A quick way to generate something like: public static final int FLAG_ONE = 1;

<u>Java, JavaScript: switch</u>
An instant switch-case block.

Java: toarray
Boilerplate for converting an ArrayList to an array[].

Java, JavaScript: try
Creates a new try-catch block. A really nice use for this is to easily surround an existing chunk of code with a try-catch block. Just highlight the code you want inside the try-catch, press Control+Space, type “try”, and press Enter. Now all the code you highlighted will be inside a new try-catch.

Java, JavaScript: while
Creates the structure of a “while” loop in one of a few different ways.

I don’t really tend to use the default shortcuts with CSS, HTML, or XML files, although these types of files are interesting because they have an option for “new file” template types. As an example, go to File Resources in DDE and click the “New File Resource” button, and create a new file called “test.html”. Now, in the empty file that opens in the HTML editor, press Control+Space. You will be given several options for boilerplate code for a new HTML file. If you single-click on any of the New HTML File options you can see a preview of the HTML that will be inserted for that option; if you double-click an option the file will be pre-filled with the boilerplate code.

Ideas for Customization
Let’s start with a simple custom Java template that generates a reference to the current database in an agent. Open the Notes Preferences dialog in DDE and navigate to Java - Editor - Templates. Click the “New” button to create a new code template and use the following values:

Name: thisdb
Context: Java
Description: add a reference to the current database to an agent
Pattern:
```
Database ${db} = $
{agentContext:localVar(lotus.domino.AgentContext)}.getCurrentDatabase();
```

Click OK to save this new template, then OK again to close the preferences dialog, then open a new Java agent and type “thisdb” followed by Control+Space. The code should auto-complete to:

```
Database db = agentContext.getCurrentDatabase();
```

You will be able to tab to and edit the “db” and the “agentContext” variables. Here’s a little more information about how the pattern in the template works:

The ${db} variable is a placeholder for an arbitrary variable name in the pattern. It will default to “db”, but immediately after clicking Control+Space you will also be able to use the tab key to navigate to it and change it to whatever you want. What’s useful about this is that if ${db} appears anywhere else in your pattern, then all occurrences of ${db} will change if you modify any one of them.

The ${agentContext:localVar(lotus.domino.AgentContext)} variable indicates that any locally defined variable (hence the “localVar” tag) of type lotus.domino.AgentContext that is within the scope at the point where the template is used will be substituted for the template variable automatically. If there are no local lotus.domino.AgentContext variables available, “agentContext” will be used as the variable name. If there are more than one local variables of that type, the one closest to the code will be used by default, but you will be able to use the arrow keys to chose amongst the multiple options when you tab to the field.

There is also an “Automatically insert” checkbox in the dialog when you create or edit the template. If this is checked, the template code will be inserted immediately after pressing Control+Space. If it’s not checked you will see what the template does after pressing Control+Shift, and then will have to either double-click or press Enter to insert the template.

For a slightly longer example, here is a template for looping through the documents in a view:

Name: viewloop
Context: Java
Description: loop through all documents in a view
Pattern:

```
View ${view} = ${db:localVar(lotus.domino.Database)}.getView("${}");
${view}.setAutoUpdate(false);
Document ${doc} = ${view}.getFirstDocument();
Document ${tempDoc} = null;
while (${doc} != null) {
   // ${todo} do stuff
   ${cursor}
   ${tempDoc} = ${view}.getNextDocument(${doc});
   ${doc}.recycle();
   ${doc} = ${tempDoc};
}
```

You can see that the ${view}, ${doc}, and ${tempDoc} variables are used in multiple places, and if you change the variable name in one place it will change for all instances of that variable. This was an option for the ${db} variable as well; even though it was defined with a localVar indicator initially, it can be reused later as ${db} if you choose.

I also used a ${cursor} variable to indicate where the cursor should end up after you have tabbed through the different variables.

Using templates in JavaScript works exactly the same way, although the “var” and “localVar” references don’t normally resolve for me (and remember: this only works for JavaScript script libraries, not SSJS editor in XPages or other places). For example:

Name: newdoc
Context: javascript
Description: create a new Notes document
Pattern:

```
var ${doc} = ${db:var}.createDocument();
${doc}.appendItemValue("Form", "${}");
${doc}.appendItemValue("${cursor}", "");
if (${doc}.save()) {

} else {

}
```

Other ideas for Java or JavaScript code templates include:

- Automatically adding logging to try-catch blocks
- Date formatting or parsing
- Reading or writing files
- Parsing an XML file or string
- Checking for both a null and an empty value in an if-then loop

The templates for CSS, HTML, and XML tend to be more along the lines of direct text insertion of common tags and such, but those might still be useful for you. And you can define “new file” options for these types of files too, so you could pre-define an HTML or a CSS file structure if you typically use DDE as an editor for that sort of thing.