Notes Java UI API Exerciser

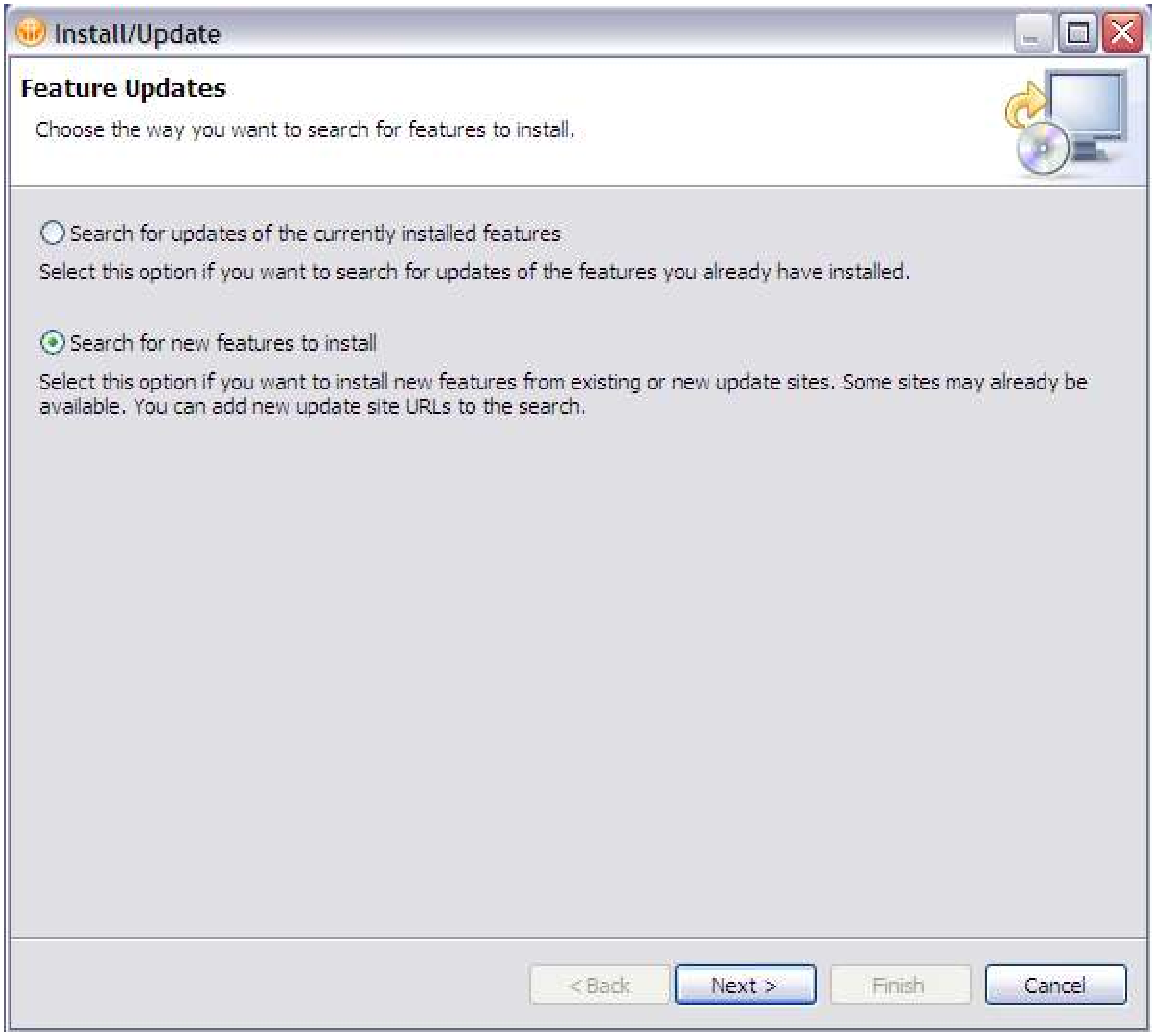
License:	Apache License v2
Last version:	1.5.2
Authors:	Ryan J Baxter (IBM), Stanton Sievers (IBM)
Platform:	Lotus Notes 8.5.2 or above

Install

Download this plug-in and extract it to local folder, such as C:\APIExerciser. Add the following line in <Notes>\framework\rcp\plugin_customization.ini to enable menu File -> Applications -> Install....

```
com.ibm.notes.branding/enable.update.ui=true
```

- Start Notes and select menu File -> Applications -> Install... Then you will see the following window. Select the second option ("Search for new features to install") and then click Next.



Feature Updates

Choose the way you want to search for features to install.

( ) Search for updates of the currently installed features

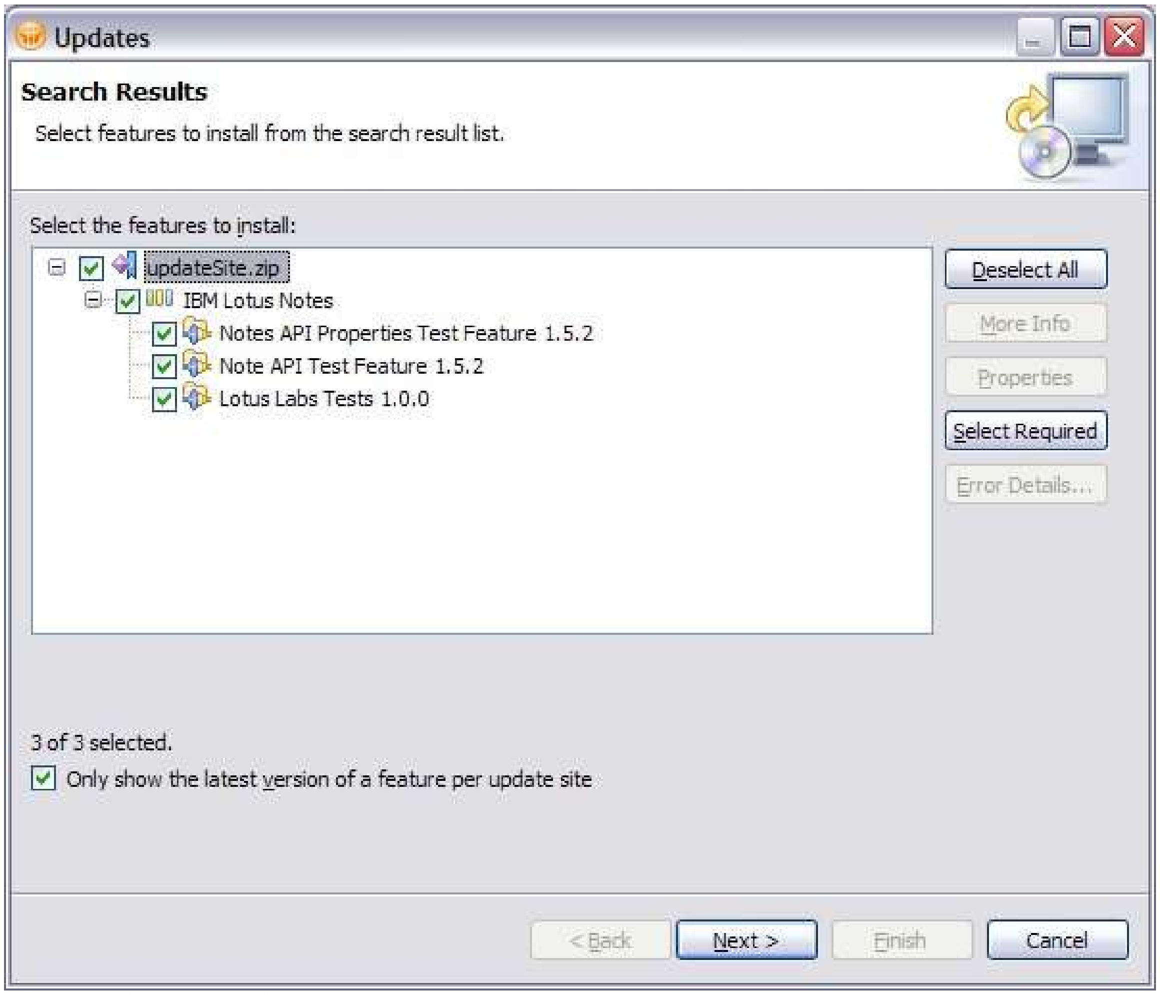
Select this option if you want to search for updates of the features you already have installed.

[✓ Search for new features to install]

Select this option if you want to install new features from existing or new update sites. Some sites may already be available. You can add new update site URLs to the search.

Figure 1 Select the second option to install new features

- Click the **Add Folder Location…** button on the Install window, and then navigate to the directory that contains the site.xml file (C:\APIExerciser\updatesite). Ensure that the “Notes API Test Feature” and the “Lotus Labs Tests” features are selected. The “Notes API Properties Test Feature” is optional.


Updates

Search Results

Select features to install from the search result list.

Select the features to install:

- [✓] updateSite.zip
  - [✓] IBM Lotus Notes
    - [✓] Notes API Properties Test Feature 1.5.2
    - [✓] Note API Test Feature 1.5.2
    - [✓] Lotus Labs Tests 1.0.0

3 of 3 selected.

[✓] Only show the latest version of a feature per update site

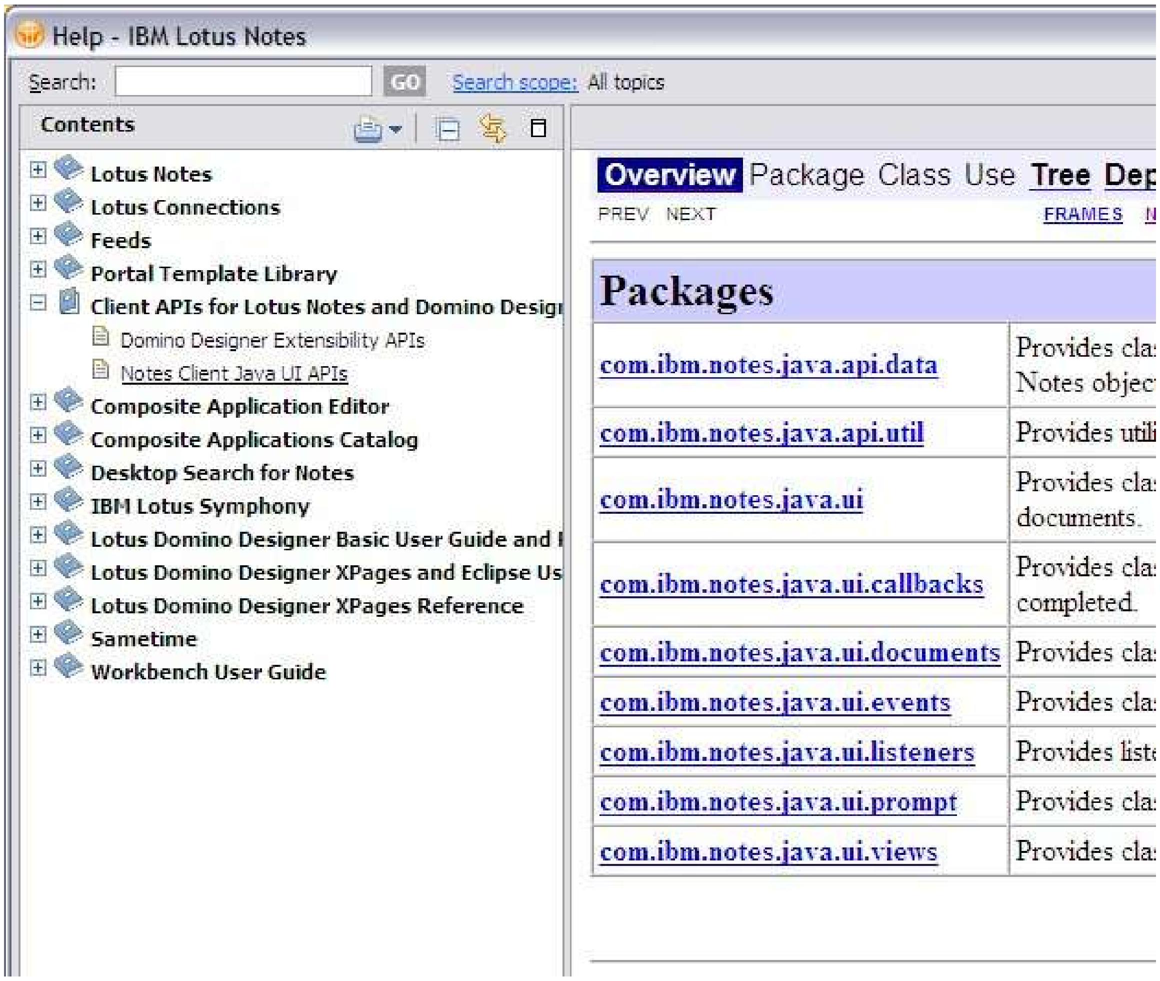
Figure 2 Select the features you want to install

- Click Finish and accept the remaining prompts.
- Once the new features are installed, you will be asked to restart Lotus Notes.

Usage

This project exemplifies the use of the Java UI APIs for Lotus Notes. It contributes a sidebar view that allows users to add current documents and views in order to view information about them and to take action on them. It also contributes several stand-alone views that allow for interaction with other functionality of the Java UI APIs, such as launching database pickers, composing documents, running agents, and opening views and framesets.

By examining the source code for this project's plugins, one will be able to gain insight into how to utilize the Java UI APIs for Lotus Notes. In order to better understand the rest of this document, one should familiarize oneself with the Java UI APIs. The 8.5.1 javadoc for the APIs can be found in the Notes help documentation, as seen in Figure 3. The newest documentation can be found on the Lotus Notes and Domino Application Development wiki along with other useful resources:



Lotus Notes
Lotus Connections
Feeds
Portal Template Library
Client APIs for Lotus Notes and Domino Desig
Domino Designer Extensibility APIs
Notes Client Java UI APIs
Composite Application Editor
Composite Applications Catalog
Desktop Search for Notes
IBM Lotus Symphony
Lotus Domino Designer Basic User Guide and
Lotus Domino Designer XPages and Eclipse Us
Lotus Domino Designer XPages Reference
Sametime
Workbench User Guide

Packages	
com.ibm.notes.java.api.data	Provides cla Notes objec
com.ibm.notes.java.api.util	Provides util
com.ibm.notes.java.ui	Provides cla documents.
com.ibm.notes.java.ui.callbacks	Provides cla completed.
com.ibm.notes.java.ui.documents	Provides cla
com.ibm.notes.java.ui.events	Provides cla
com.ibm.notes.java.ui.listeners	Provides liste
com.ibm.notes.java.ui.prompt	Provides cla
com.ibm.notes.java.ui.views	Provides cla

Figure 3 Location of the API javadoc in the Notes help

The Java UI APIs also make use of some Java classes from Notes.jar. The documentation for these classes may be found in the Lotus Notes Help in the “Lotus Domino Designer Basic User Guide and Reference” section under “Java/CORBA Classes.”

Notes UI Elements Sidebar

The Notes UI Elements Sidebar allows users to add the current Notes document, view, or element to the sidebar in order to view information about it or take action upon it. To add the current NotesUIElement to the sidebar, select “Add Element” to add either the current element – either a view or document. To add a specific NotesUIElement to the sidebar, simply give focus to a document or view in the Notes UI and select “Add Document” or “Add View” in the sidebar to add a document or a view, respectively.

One may also “Add Data” which adapts the current NotesUIElement to the applicable data classes using the adapter pattern. For example, given a view this will adapt the view to NotesViewData and NotesDatabaseData and display it in the sidebar.

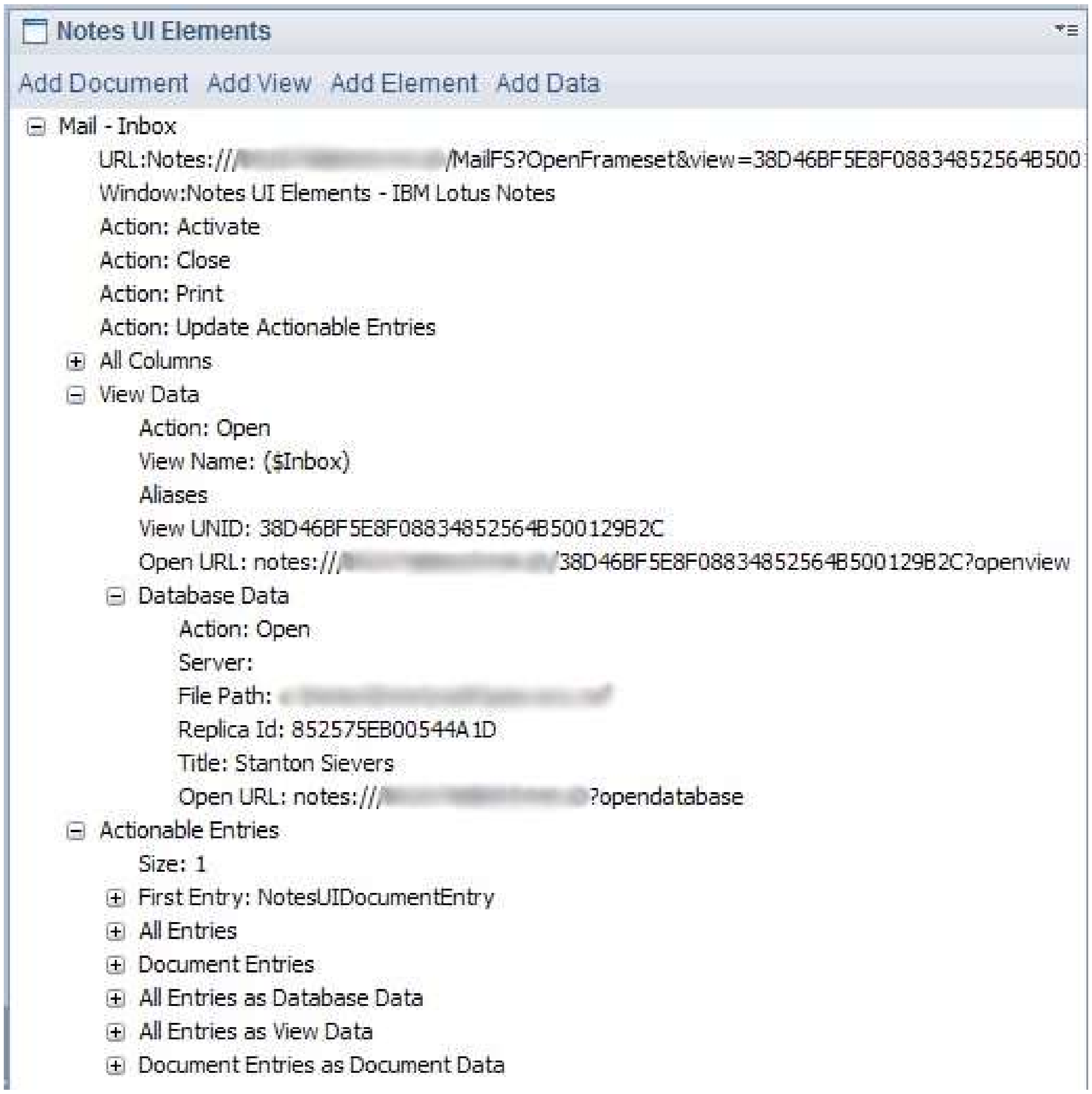
Information in the sidebar is represented in two ways. First, properties of the view or document are displayed in the format “Property: Property Value”. For instance, in Figure 4 one can find “URL: Notes:///....” This denotes the url of the view.

Another way that information is displayed is as an action. Actions are always displayed in the form “Action: Some action”. Actions may be double-clicked in order to execute them. In Figure 4, double-clicking “Action: Close” will close the view using the NotesUIView.close() method.

NotesUIElements in the sidebar may also have subtrees which themselves have properties and actions. In Figure 4 one can see an “All Columns” subtree that can be expanded to view each column information for the view.

NotesUIView

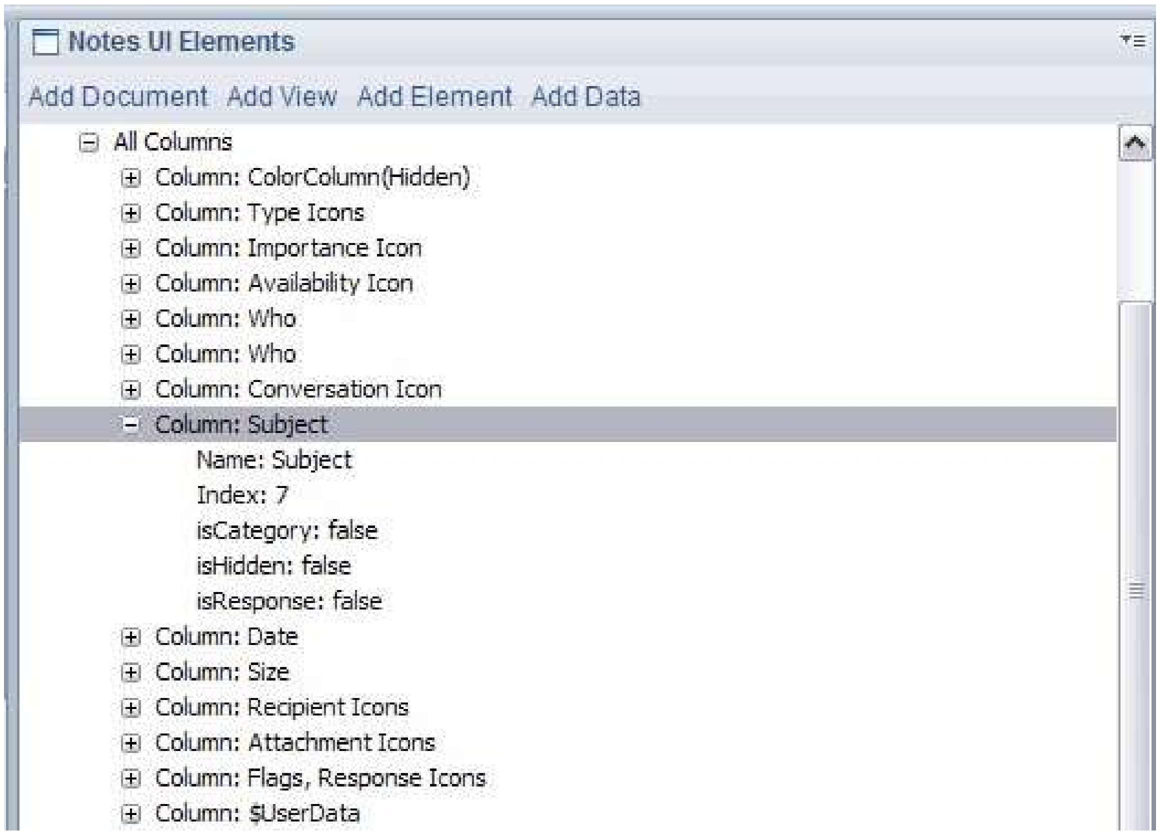
In Notes 8.5.2, new Java UI APIs were introduced to give consumers more information about the current Notes view. In 8.5.1 the NotesUIView APIs were limited but in 8.5.2 new APIs have been added to get column information from the view as well as to get the actionable entries in the view.



```
⊟ Mail - Inbox
    URL:Notes://[redacted]/MailFS?OpenFrameset&view=38D46BF5E8F08834852564B500
    Window:Notes UI Elements - IBM Lotus Notes
    Action: Activate
    Action: Close
    Action: Print
    Action: Update Actionable Entries
  ⊞ All Columns
  ⊟ View Data
      Action: Open
      View Name: ($Inbox)
      Aliases
      View UNID: 38D46BF5E8F08834852564B500129B2C
      Open URL: notes://[redacted]/38D46BF5E8F08834852564B500129B2C?openview
    ⊟ Database Data
        Action: Open
        Server:
        File Path: [redacted]
        Replica Id: 852575EB00544A1D
        Title: Stanton Sievers
        Open URL: notes://[redacted]?opendatabase
  ⊟ Actionable Entries
      Size: 1
    ⊞ First Entry: NotesUIDocumentEntry
    ⊞ All Entries
    ⊞ Document Entries
    ⊞ All Entries as Database Data
    ⊞ All Entries as View Data
    ⊞ Document Entries as Document Data
```

Figure 4 An example NotesUIView in the sidebar

Figure 5 exhibits more information regarding the columns of a view. Each column represents a NotesUIViewColumn. Figure 5 shows the “Subject” column of Notes mail file. The entire list of columns is attained from the view using the NotesUIView.getColumns() method.


Notes UI Elements

Add Document Add View Add Element Add Data

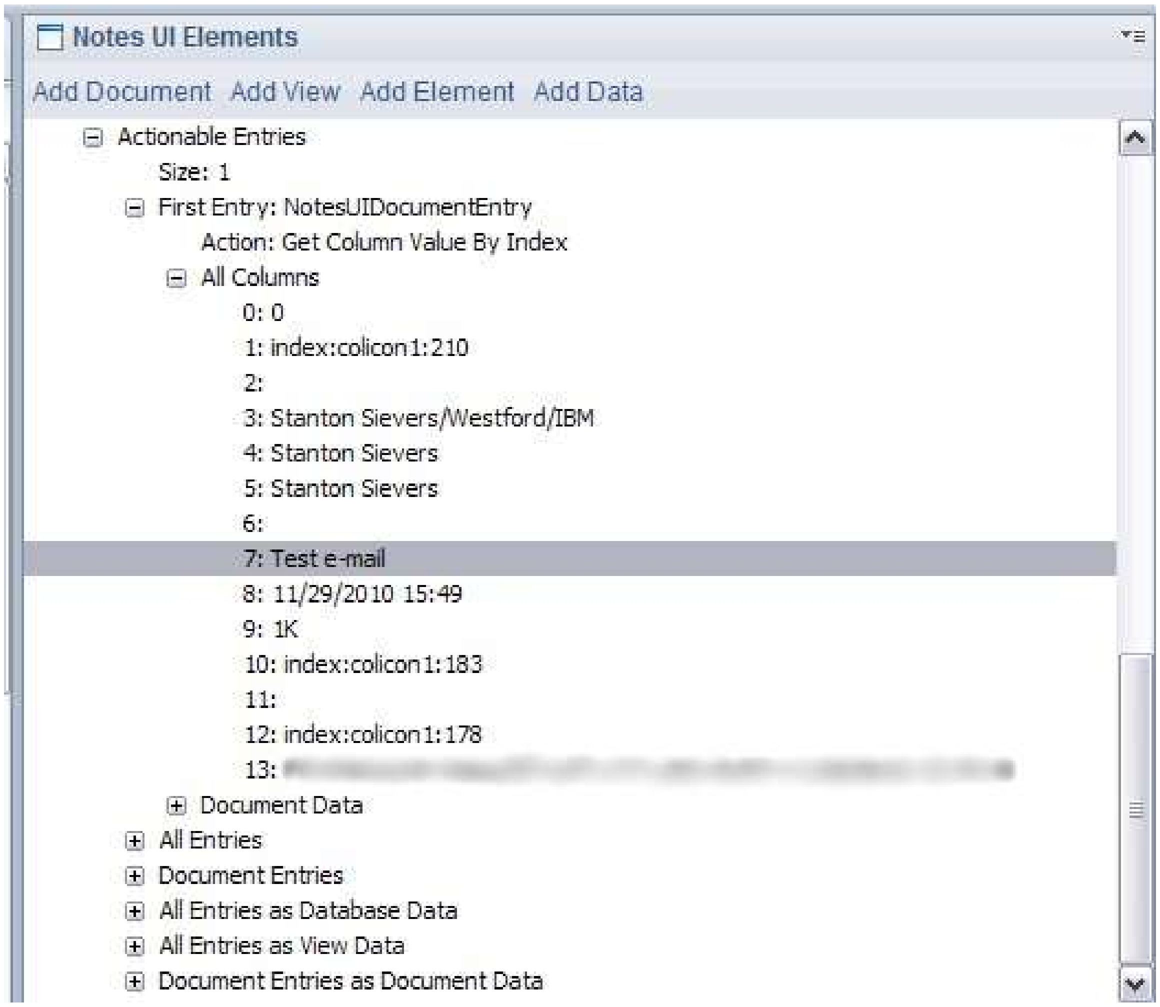
```
⊟ All Columns
    ⊞ Column: ColorColumn(Hidden)
    ⊞ Column: Type Icons
    ⊞ Column: Importance Icon
    ⊞ Column: Availability Icon
    ⊞ Column: Who
    ⊞ Column: Who
    ⊞ Column: Conversation Icon
    ⊟ Column: Subject
          Name: Subject
          Index: 7
          isCategory: false
          isHidden: false
          isResponse: false
    ⊞ Column: Date
    ⊞ Column: Size
    ⊞ Column: Recipient Icons
    ⊞ Column: Attachment Icons
    ⊞ Column: Flags, Response Icons
    ⊞ Column: $UserData
```

Figure 5 An example "All Columns" subtree of a NotesUIView

Figure 6 illustrates the “Actionable Entries” of the view which are obtained through the NotesUIView.getActionableEntries() method. In the simplest case, the actionable entries are simply the selected entries in the view; however, in some cases this is not true and the API documentation explains these cases more thoroughly.

For each actionable entry, the column values can be obtained through the “All Columns” view. The columns correspond to the NotesUIViewColumns from Figure 5. Figure 6 shows the column value for this entry that corresponds to the Subject column of the view.

Through this sidebar UI one may examine all of the actionable entries, only the actionable entries which are NotesUIDocumentEntries, or the actionable entries adapted to specific data objects using the adapter pattern.

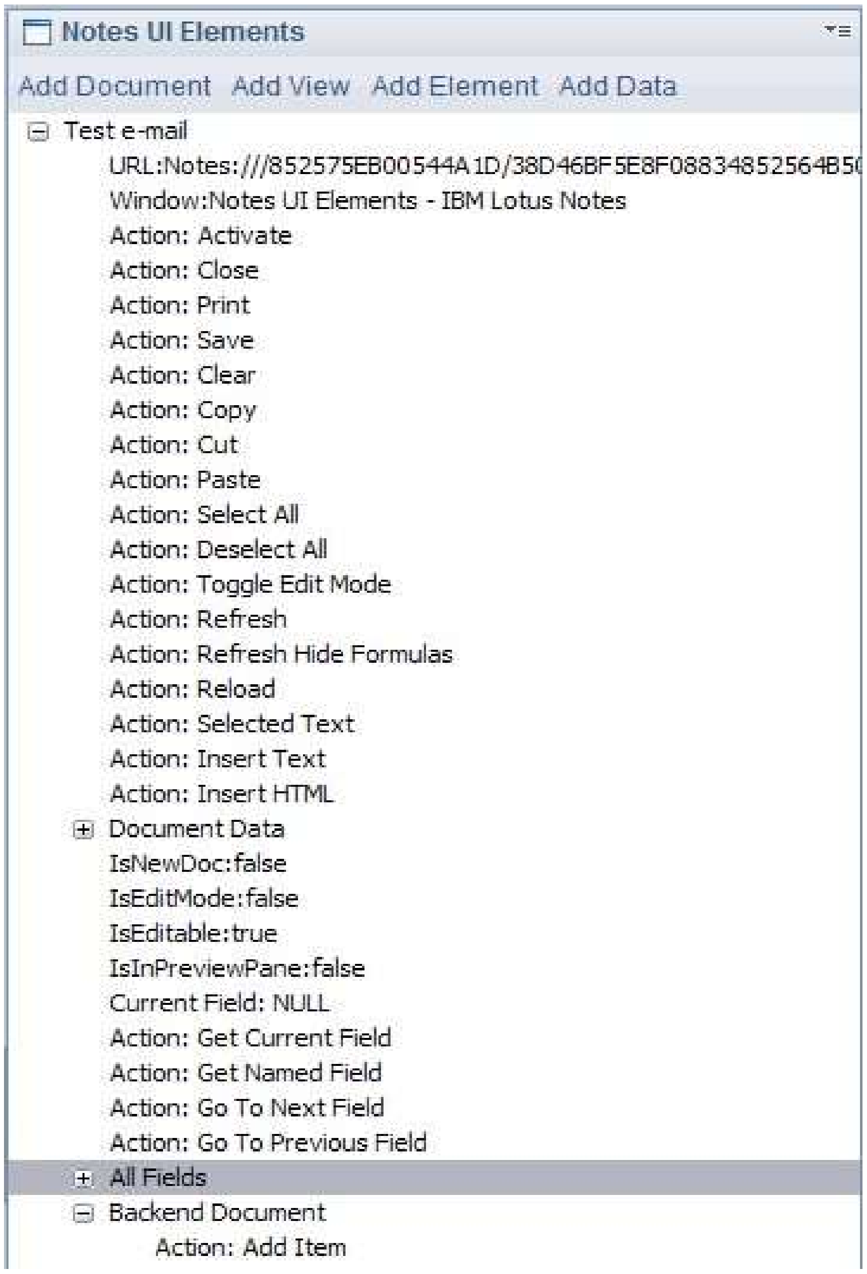


```
⊟ Actionable Entries
    Size: 1
  ⊟ First Entry: NotesUIDocumentEntry
      Action: Get Column Value By Index
    ⊟ All Columns
        0: 0
        1: index:colicon1:210
        2:
        3: Stanton Sievers/Westford/IBM
        4: Stanton Sievers
        5: Stanton Sievers
        6:
        7: Test e-mail
        8: 11/29/2010 15:49
        9: 1K
        10: index:colicon1:183
        11:
        12: index:colicon1:178
        13: [redacted]
    ⊞ Document Data
  ⊞ All Entries
  ⊞ Document Entries
  ⊞ All Entries as Database Data
  ⊞ All Entries as View Data
  ⊞ Document Entries as Document Data
```

Figure 6 An example "Actionable Entries" subtree of a NotesUIView

NotesUIDocument

The document APIs have remained unchanged since 8.5.1. Figure 7 shows the familiar properties and actions that may be taken on a document.


Notes UI Elements

Add Document   Add View   Add Element   Add Data

```
⊟ Test e-mail
      URL:Notes:///852575EB00544A1D/38D46BF5E8F08834852564B5
      Window:Notes UI Elements - IBM Lotus Notes
      Action: Activate
      Action: Close
      Action: Print
      Action: Save
      Action: Clear
      Action: Copy
      Action: Cut
      Action: Paste
      Action: Select All
      Action: Deselect All
      Action: Toggle Edit Mode
      Action: Refresh
      Action: Refresh Hide Formulas
      Action: Reload
      Action: Selected Text
      Action: Insert Text
      Action: Insert HTML
   ⊞ Document Data
      IsNewDoc:false
      IsEditMode:false
      IsEditable:true
      IsInPreviewPane:false
      Current Field: NULL
      Action: Get Current Field
      Action: Get Named Field
      Action: Go To Next Field
      Action: Go To Previous Field
   ⊞ All Fields
   ⊟ Backend Document
         Action: Add Item
```

Figure 7 An example NotesUIDocument in the sidebar

Documents in the sidebar also allow the ability to view information about the NotesBEDocument. There is simply one action for the backend document: Add Item. This allows items from the backend to be retrieved by name and added to the sidebar on-demand. The items are added under the “Backend Document” subtree.

Because documents contain a collection of NotesUIFields, they are also listed as a subtree of a document. Each field entry in the sidebar displays the name of the field and allows one to take actions upon the fields, as can be seen in Figure 8.

Top-level items in the sidebar can be refreshed or removed by right-clicking on them and selecting the corresponding action from the context menu. Top-level items will automatically be removed based on the functionality of their close listeners. For example, if a view is in the sidebar and the tab in which the view resides is closed, the view will be removed from the sidebar. Documents will be removed in the same manner.

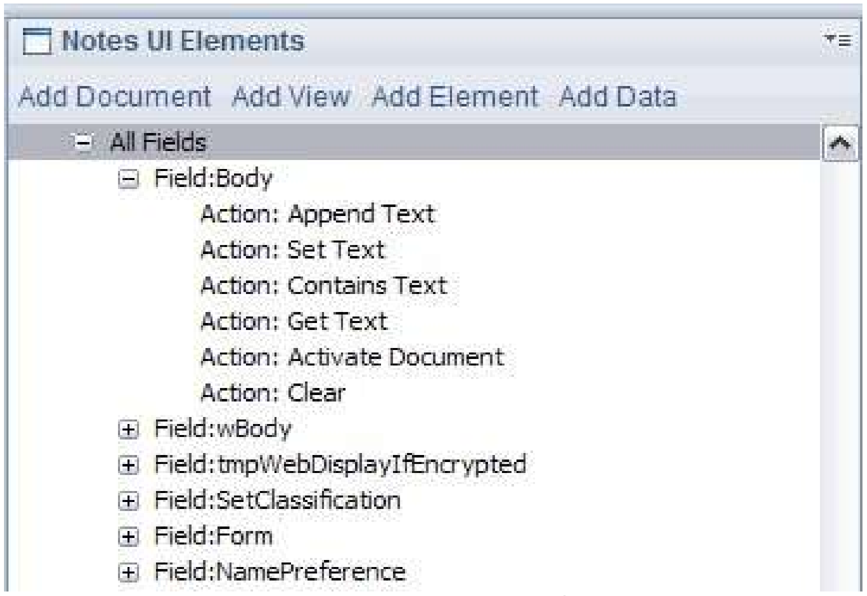


Figure 8 An example "All Fields" subtree of a NotesUIDocument

Lotus Labs Tests

Another aspect of this project is the Lotus Labs Tests perspective which displays several views that offer functionality based on the API. To get to it, find the Lotus Labs Tests entry in the “Open” menu.

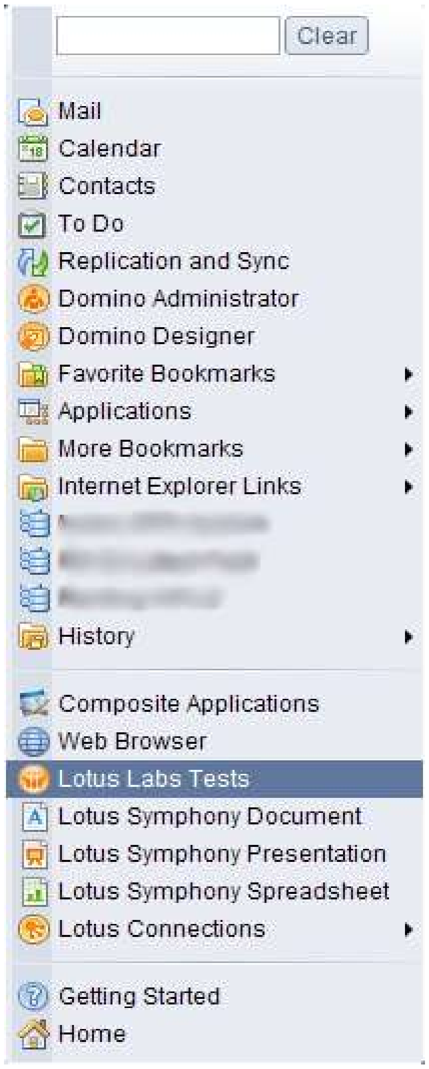


Figure 9 Lotus Labs Tests in the Notes Open menu

The Lotus Labs Tests tab shows a table that lists all of the other views that can be opened. To open one of these views, simply double-click it.

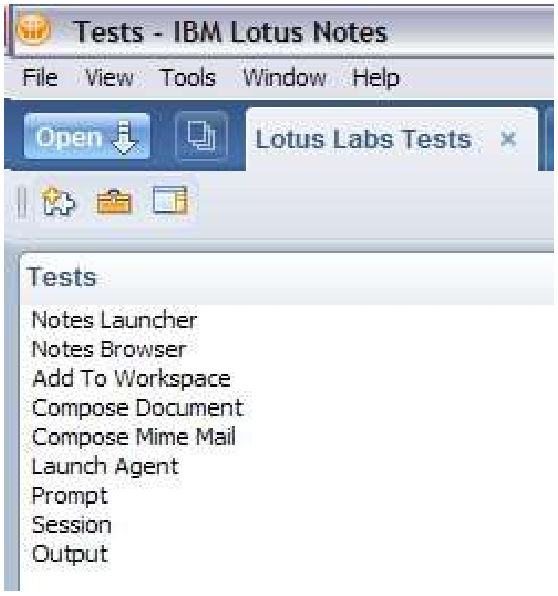


Figure 10 The Lotus Labs Tests tab

Notes Launcher

The Notes Launcher allows the user to open pages, framesets, and views in a given database using the following API functions:

```
com.ibm.notes.java.ui.NotesUIWorkspace.openPage(NotesPageData)
com.ibm.notes.java.ui.NotesUIWorkspace.openFrameset(NotesFramesetData)
com.ibm.notes.java.ui.NotesUIWorkspace.openView(NotesViewData)
```

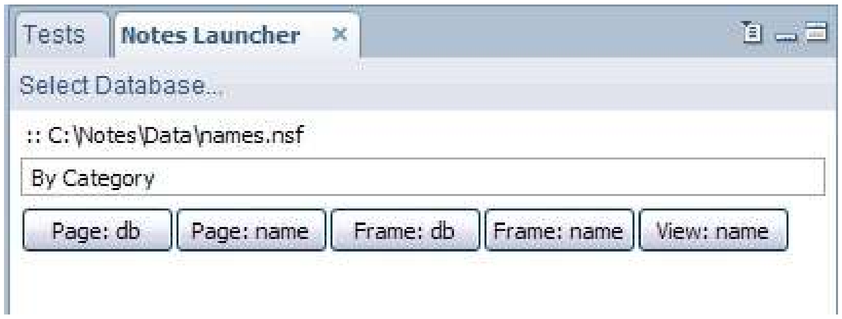


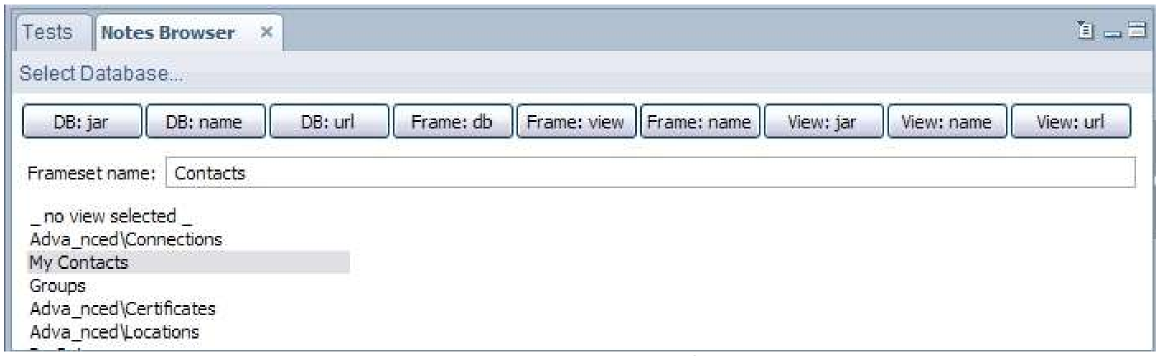
Figure 11 Using Notes Launcher to open the “By Category” view in names.nsf

In the past, each button in Notes Launcher attempted to open the given element in one of two ways. The “name” suffix denoted that the element was to be opened with only the given data (i.e. the name) while the “db” suffix denoted that it was to be opened with more data retrieved from the database in a NotesSessionJob. Currently, they open the element with the same data; however, “db” still opens it in a NotesSessionJob while “name” opens it on the UI thread. In future releases, this behavior will be cleaned up to more accurately reflect the current state of the APIs.

Notes Browser

The Notes Browser's functionality is similar to the Notes Launcher, but the implementations are different. Notes Browser exercises these API functions:

```
com.ibm.notes.java.ui.NotesUIWorkspace.openDatabase(NotesDatabaseData)
com.ibm.notes.java.ui.NotesUIWorkspace.openUrl(String)
com.ibm.notes.java.ui.NotesUIWorkspace.openFrameset(NotesFramesetData)
com.ibm.notes.java.ui.NotesUIWorkspace.openFrameset(NotesFramesetData,
NotesViewData)
com.ibm.notes.java.ui.NotesUIWorkspace.openFrameset(NotesFramesetData,
String)
com.ibm.notes.java.ui.NotesUIWorkspace.openView(NotesViewData)
```



Frameset name: Contacts

_ no view selected _
Adva_nced\Connections
My Contacts
Groups
Adva_nced\Certificates
Adva_nced\Locations

Figure 12 Using Notes Browser to browse names.nsf

The key difference between Notes Launcher and Notes Browser is that the Notes Browser lists all of the views in the database, as seen in Figure 10. In this way, one can simply select a view to open, instead of having to type a view name.

Each button in the Notes Browser attempts to open the database element in one of three ways. The "jar" suffix indicates that the element will be opened by constructing its corresponding data with a Notes.jar class (i.e. View or Database). The "url" suffix indicates that the element will be opened via the openUrl(String) method mentioned above. The "name" suffix indicates that the name of the element will be used when constructing the corresponding data and opening the element, except in the case of "Frame: name," where the view name is used. "Frame: view" utilizes the openFrameset (NotesFramesetData, NotesViewData) method mentioned above to open the frameset.

Add To Workspace

The Add To Workspace view simply exercises this API function:

```
com.ibm.notes.java.ui.NotesUIWorkspace.addDatabase(NotesDatabaseData)
```

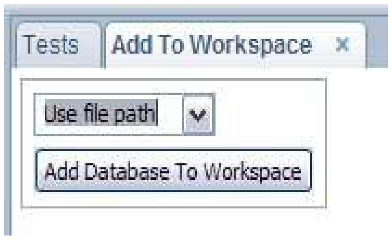


Figure 13 Add a database to the Notes workspace using one of three options

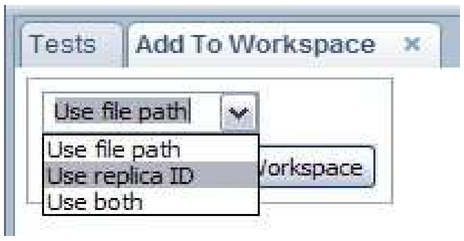


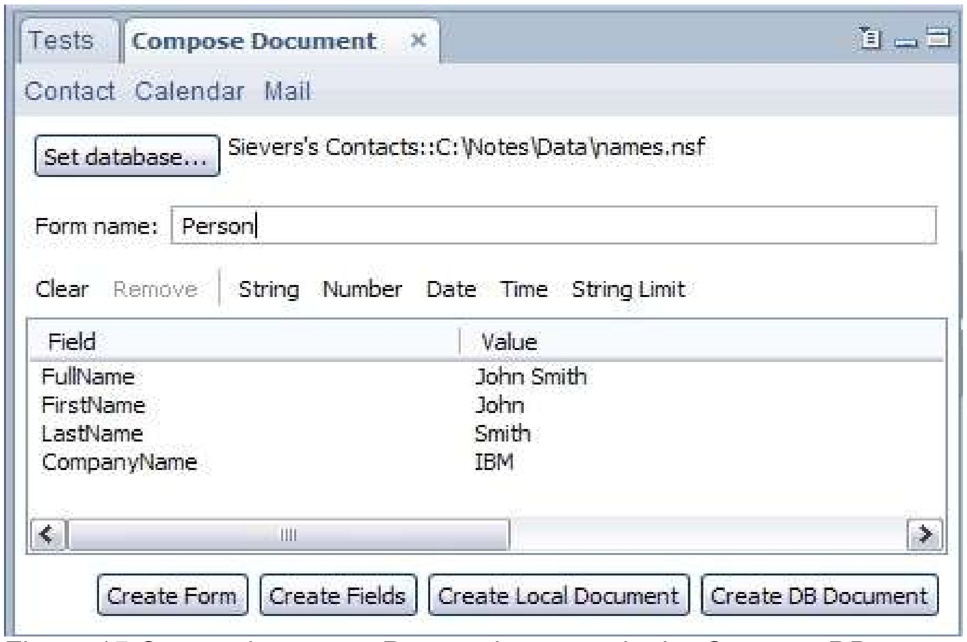
Figure 14 Three options to choose from which to choose

If the database selected does not exist in the workspace, it will be added; otherwise, it will be selected in the workspace. The drop-down box allows one to dictate what information will be used when constructing the NotesDatabaseData that is passed to the addDatabase method.

Compose Document

The Compose Document view allows documents to be composed using the following API functions:

```
com.ibm.notes.java.ui.NotesUIWorkspace.composeDocument(NotesFormData)
com.ibm.notes.java.ui.NotesUIWorkspace.composeDocument
(NotesDatabaseData, Document)
com.ibm.notes.java.ui.NotesUIWorkspace.composeDocument(Document)
```



Sievers's Contacts::C:\Notes\Data\names.nsf

Form name: Person

Field	Value
FullName	John Smith
FirstName	John
LastName	Smith
CompanyName	IBM

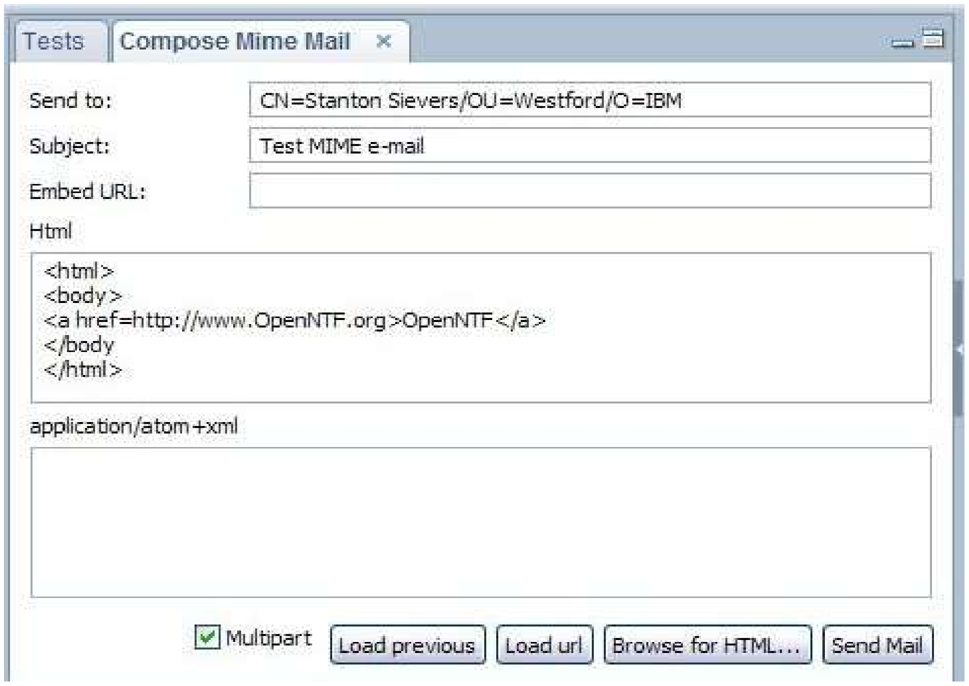
Figure 15 Composing a new Person document in the Contacts DB

The “Create Form” and “Create Fields” buttons compose a document using only NotesFormData, while “Create Local Document” and “Create DB Document” compose a document using a Document object. In the case of “Create Local Document”, NotesDatabaseData is used as well.

The “Contact”, “Calendar”, and “Mail” buttons in the view prepopulate the form name and fields in order to make composition more expedient; however, one can compose a document on any number of databases with as many fields as one chooses.

Compose Mime Mail

Much like Compose Document, Compose Mime Mail composes a document; however, in this case the document is always a mail document in the current user's mail db. Urls can be loaded to populate the body or one can browse for an html file. By selecting “Multipart”, one can decide whether to send the message as a multipart mime message or as plain html.



Send to: CN=Stanton Sievers/OU=Westford/O=IBM
Subject: Test MIME e-mail
Embed URL:
Html

```
<html>
<body>
<a href=http://www.OpenNTF.org>OpenNTF</a>
</body
</html>
```

application/atom+xml

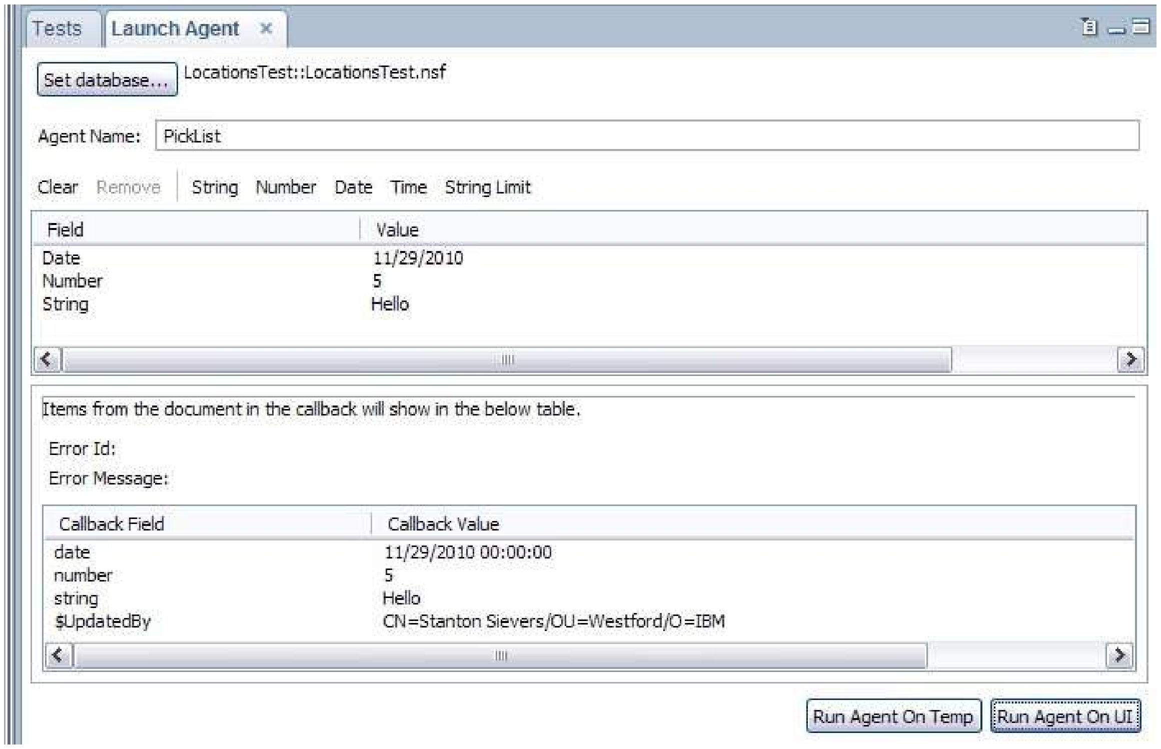
[✓ Multipart]

Figure 16 Composing a mime mail message with simple HTML

Launch Agent

The Launch Agent view allows users to run arbitrary agents in a database using this API function:

`com.ibm.notes.java.ui.NotesUIWorkspace.runAgent(NotesAgentData, NotesDocumentDataCallback, boolean)`



LocationsTest::LocationsTest.nsf

Agent Name: PickList

Field	Value
Date	11/29/2010
Number	5
String	Hello

Items from the document in the callback will show in the below table.

Error Id:

Error Message:

Callback Field	Callback Value
date	11/29/2010 00:00:00
number	5
string	Hello
$UpdatedBy	CN=Stanton Sievers/OU=Westford/O=IBM

Figure 17 Running an agent named "PickList" with sample fields

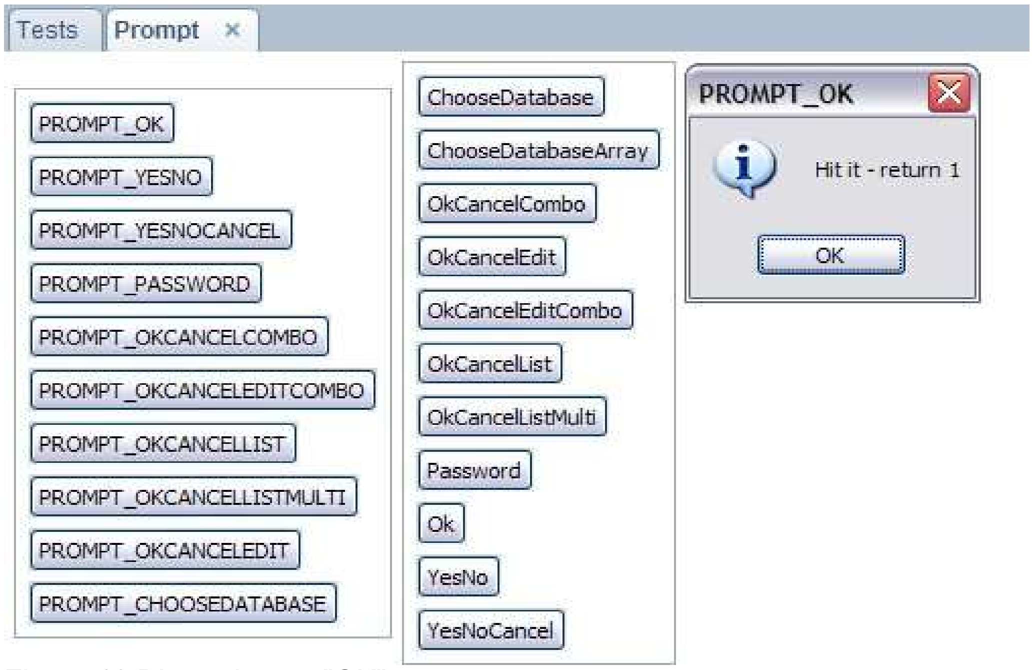
There are two options when running the agent: Run Agent On Temp and Run Agent On UI. The difference between the two is the boolean that is passed to the runAgent method. The former passes false, while the latter passes true.

The values in the callback area display field values in the document represented by the NotesDocumentData that may be retrieved from the event in the NotesDocumentDataCallback.done(NotesDocumentDataEvent) method.

Prompt

The Prompt view allows the user to display a variety of prompts using the following API functions, as well as API functions in the com.ibm.notes.java.ui.prompt.Prompt class:

`com.ibm.notes.java.ui.NotesUIWorkspace.prompt(int, String, String, Object, String[])`



PROMPT_OK

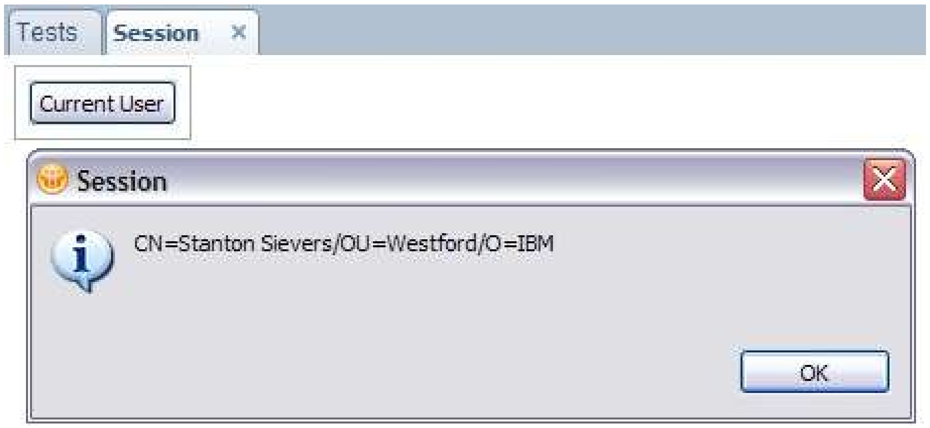
Hit it - return 1

Figure 18 Displaying an "OK" prompt

In this view, buttons prefixed with “PROMPT_” utilize the NotesUIWorkspace.prompt() method mentioned above, while the others directly use API functions in the Prompt class.

Session

The Session view queries the current user and displays it using the Session.getUserName() function in Notes.jar. The novel part of the Session view is that it extends NotesSessionJob as an inner class to accomplish this. It is a good example of how one could extend NotesSessionJob to further customize its behavior.



Session

CN=Stanton Sievers/OU=Westford/O=IBM

Figure 19 Message showing the session's current user

Output

The Output view is part of the test framework and is simply used to display output from the other views in certain scenarios.

API Properties

The Notes API Properties Test plugin demonstrates one way in which the API's property testers can be leveraged. While the classes in the src folder of the plugin may interest some who want to generate plugin.xml, the most interesting piece is the actual plugin.xml file that contributes context menus based on the results of the API's property testers. Each action in the menu will be enabled or disabled based on the property it is testing. Full documentation for all of the property testers provided by the Java UI API can be found on the Lotus Notes and Domino Application Development wiki in the javadoc for the com.ibm.notes.java.ui.internal.properties package.

Figure 20 demonstrates the Document property tester. “editmode (v:false)” relates to this expression in the plugin.xml:

```
<test property="com.ibm.notes.java.ui.document.editmode" value="false"/>
```

In this case, the document is not in edit mode, so this test returns true.

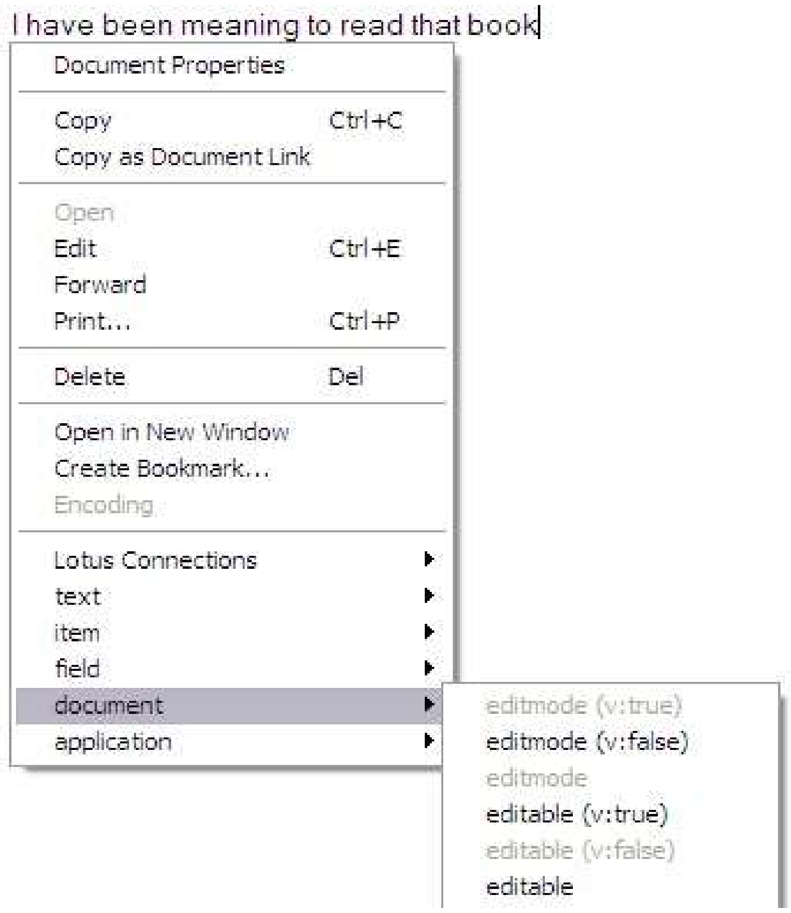


Figure 20 The document is not in edit mode but it is editable

Figure 21 gives another example, this time using selected text. In this case, the Text property tester is being used. This property tester works on the selected text. In this case “find (v:book)” relates to this expression in the plugin.xml:

```
<test property="com.ibm.notes.java.ui.text.find" value="book"/>
```

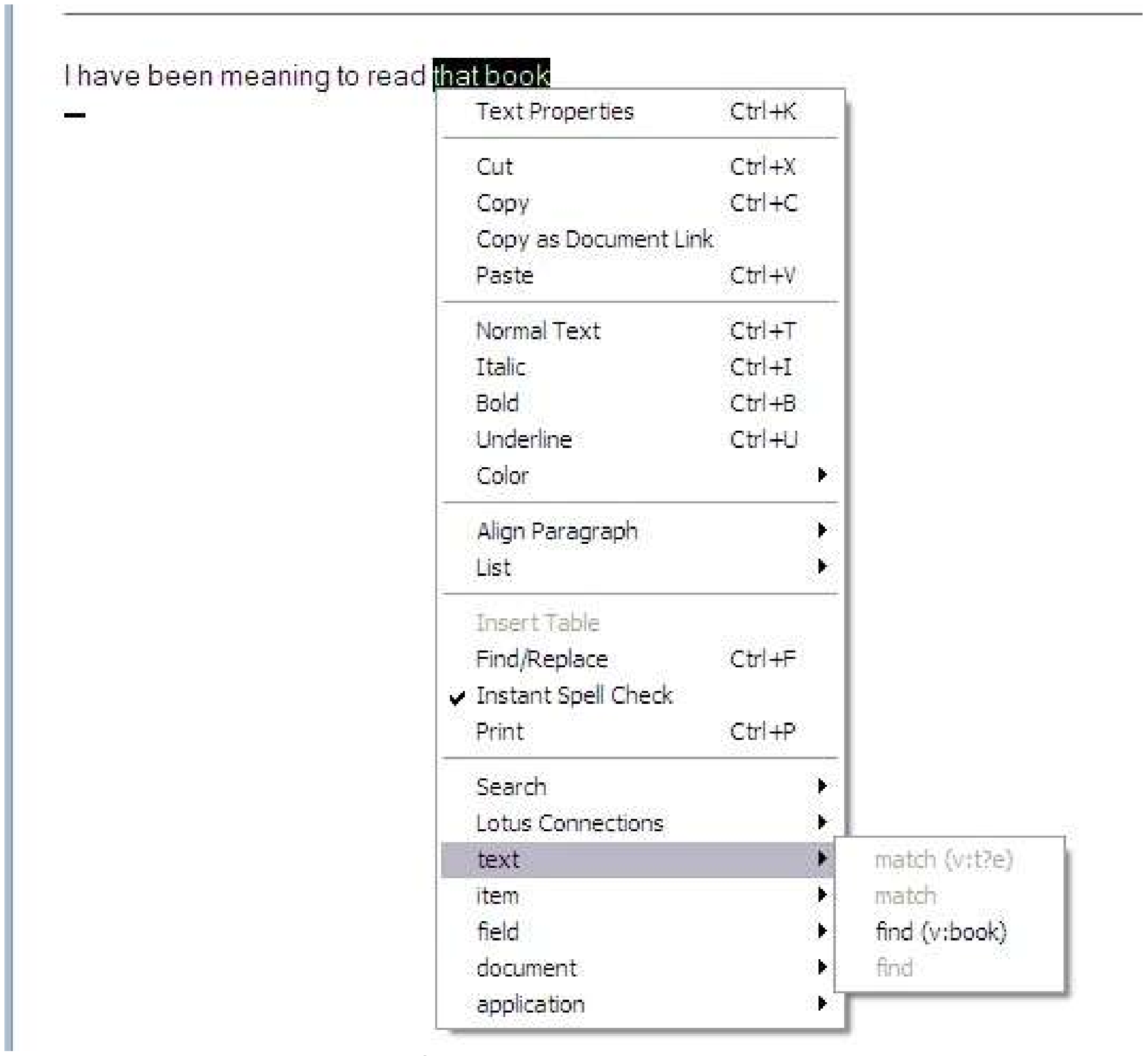

I have been meaning to read that book

Figure 21 The word "book" was found in the selected text

These are just two examples using the property testers. They can be used to control many aspects of the Lotus Notes UI. This website provides some examples in the context of Eclipse: http://wiki.eclipse.org/Command_Core_Expressions