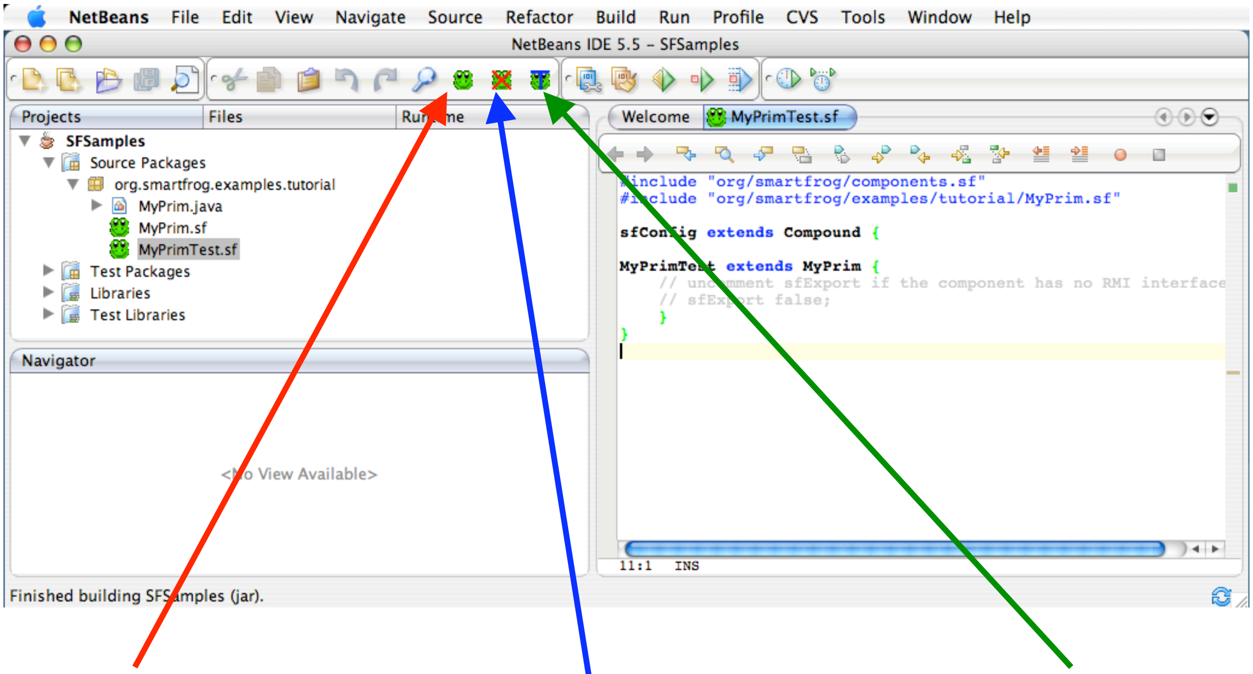
SmartFrog NetBeans Plugin

Runtime Operations

The SmartFrog NetBeans Plugin provides the following functionality as represented in the example screenshots.



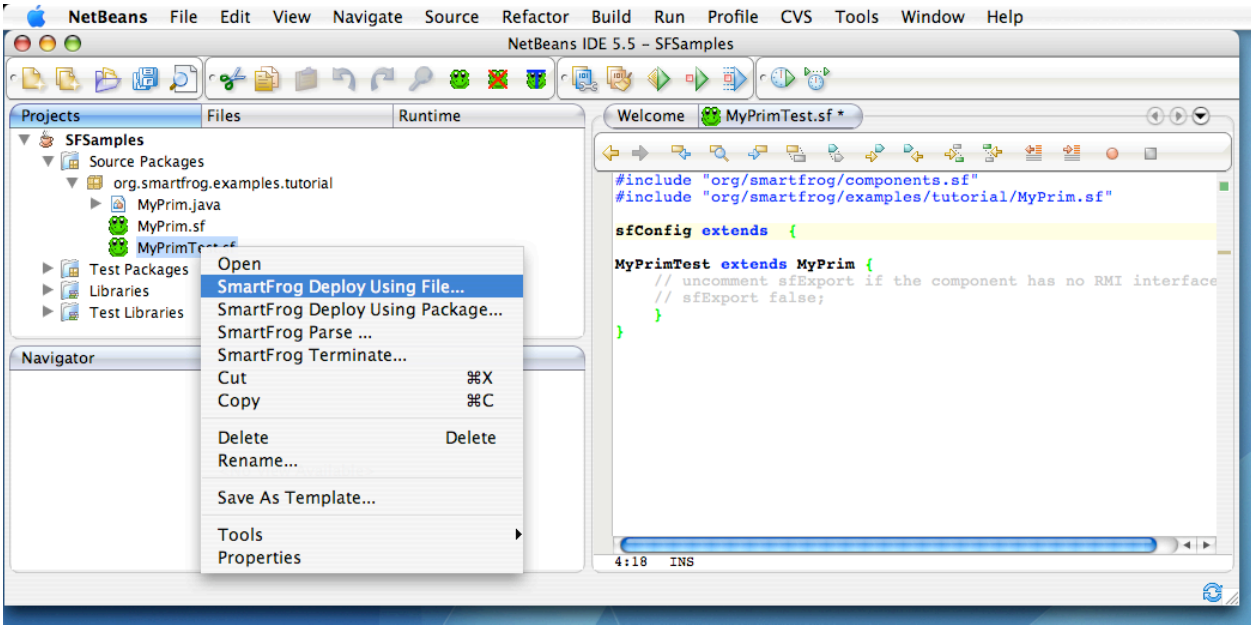
```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/examples/tutorial/MyPrim.sf"

sfConfig extends Compound {

MyPrimTest extends MyPrim {
    // uncomment sfExport if the component has no RMI interface
    // sfExport false;
    }
}
```

Finished building SFSamples (jar).

Launch SmartFrog Daemon Process *Kill SmartFrog Daemon Process* *Terminate Component*

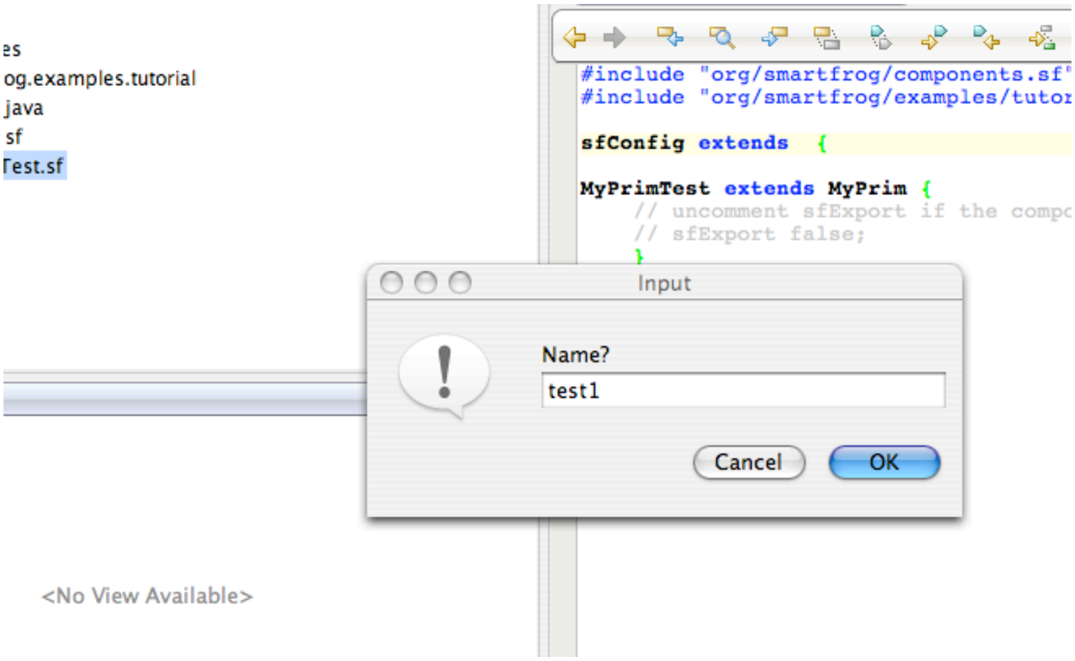
From the NetBeans UI, a developer can start and stop the SmartFrog daemon process. The developer can also terminate the named active components in the SmartFrog Daemon.

A developer can also deploy a component to the local SmartFrog daemon process by right clicking on a SmartFrog file in the project and selecting to either deploying the component directly from the file or deploy the component using a package jar file where the component description has been stored from a prior build. The following figure shows this context menu.



```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/examples/tutorial/MyPrim.sf"

sfConfig extends  {

MyPrimTest extends MyPrim {
    // uncomment sfExport if the component has no RMI interface
    // sfExport false;
    }
}
```

After making this selection the IDE will ask for the name with which you wish to use when deploying the component to SmartFrog as show in the following Diagram.



```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/examples/tutor

sfConfig extends  {

MyPrimTest extends MyPrim {
    // uncomment sfExport if the compo
    // sfExport false;
    }
```

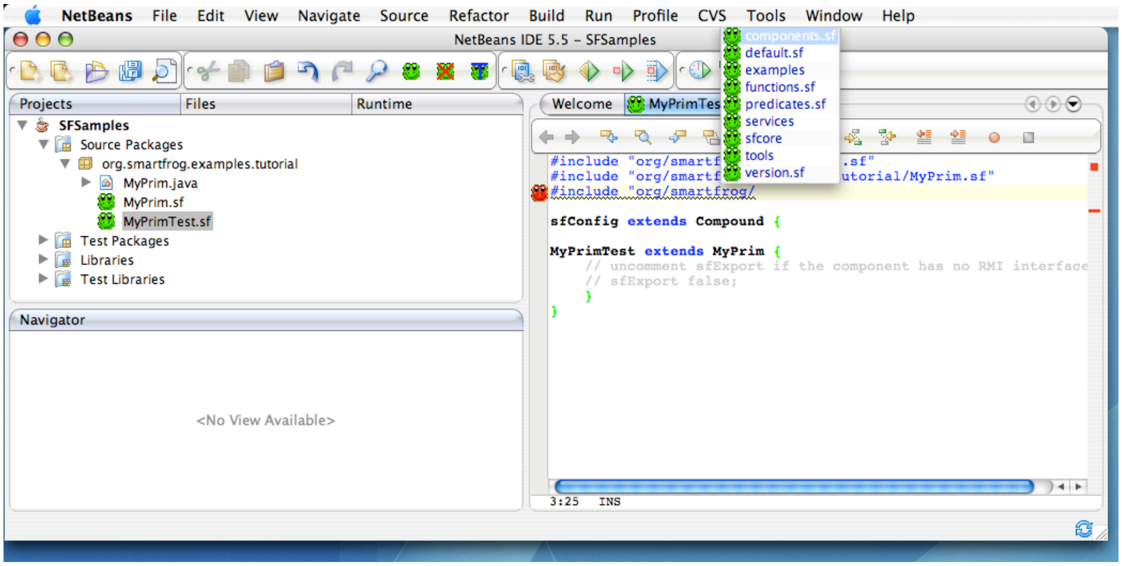
Name? test1

A developer from the context menu can also invoke a full SmartFrog parse operation displaying full verbose debug information from the SmartFrog parser.

Editor Extensions

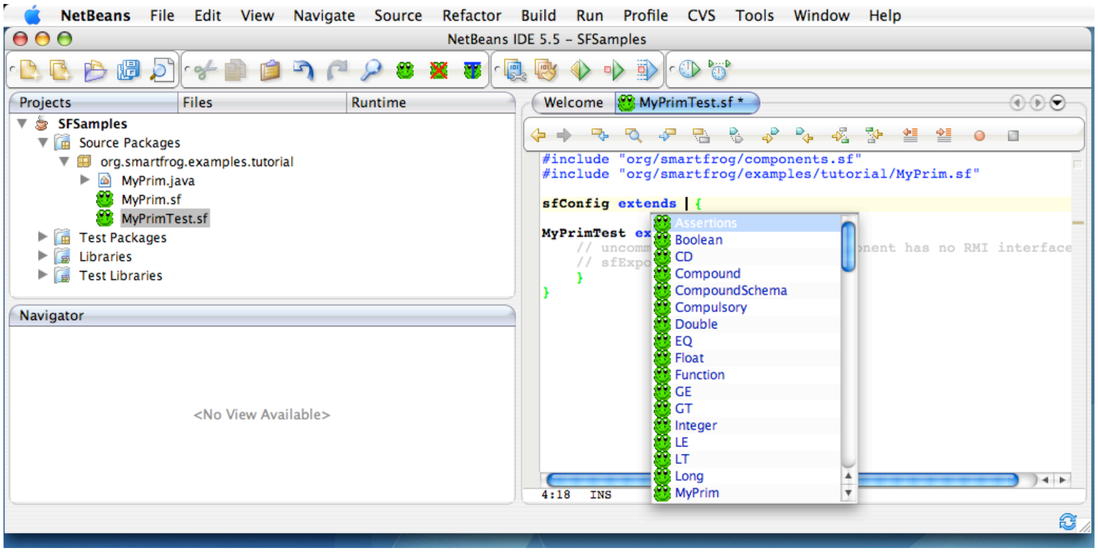
The NetBeans editor is also extended providing the following functionality.

1) **Include file auto completion** as shown in the following figure:



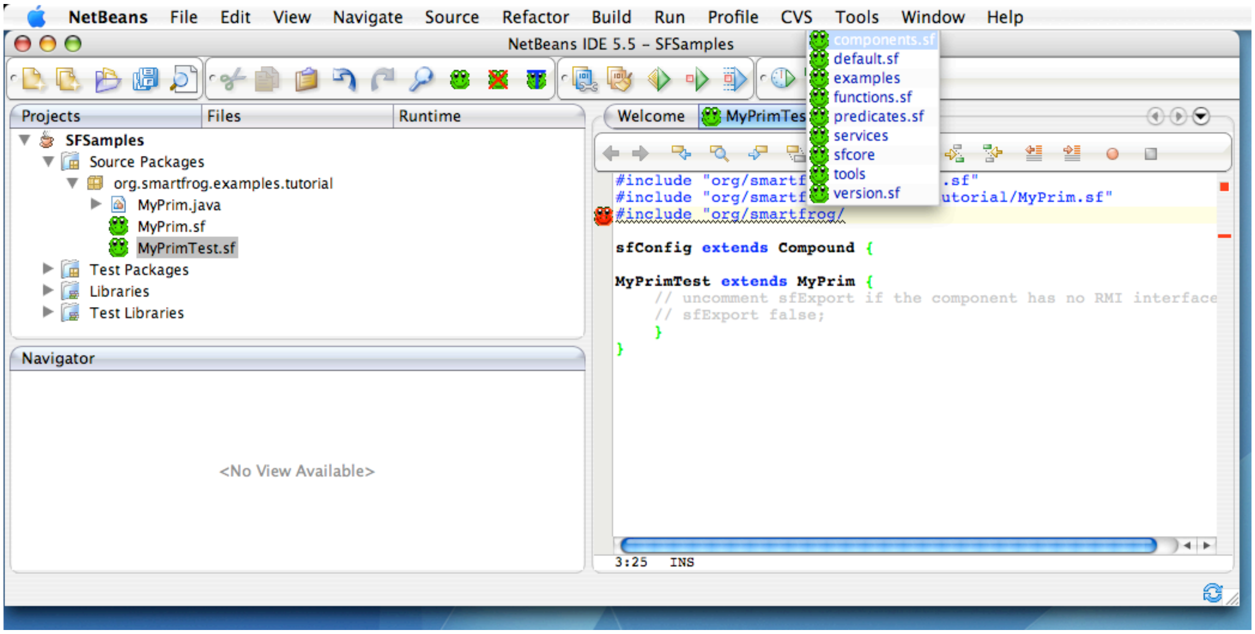
```
#include "org/smartf                .sf"
#include "org/smartf                utorial/MyPrim.sf"
#include "org/smartfrog/

sfConfig extends Compound {

MyPrimTest extends MyPrim {
    // uncomment sfExport if the component has no RMI interface
    // sfExport false;
    }
}
```

2) **Extends keyword auto completion** as show in the following figure:



```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/examples/tutorial/MyPrim.sf"

sfConfig extends | {

MyPrimTest ex                          nent has no RMI interface
    // uncomm
    // sfExpo
    }
}
```

3) **Color based syntax highlighting**. The editor will highlight the syntax using different colors. The colors for each token can be defined by the developer in the Options panel in the "Fonts and Colors" section. These are assigned for the SmartFrog language selection. The following figure shows this color based syntax as well as the Line based error indications described next.

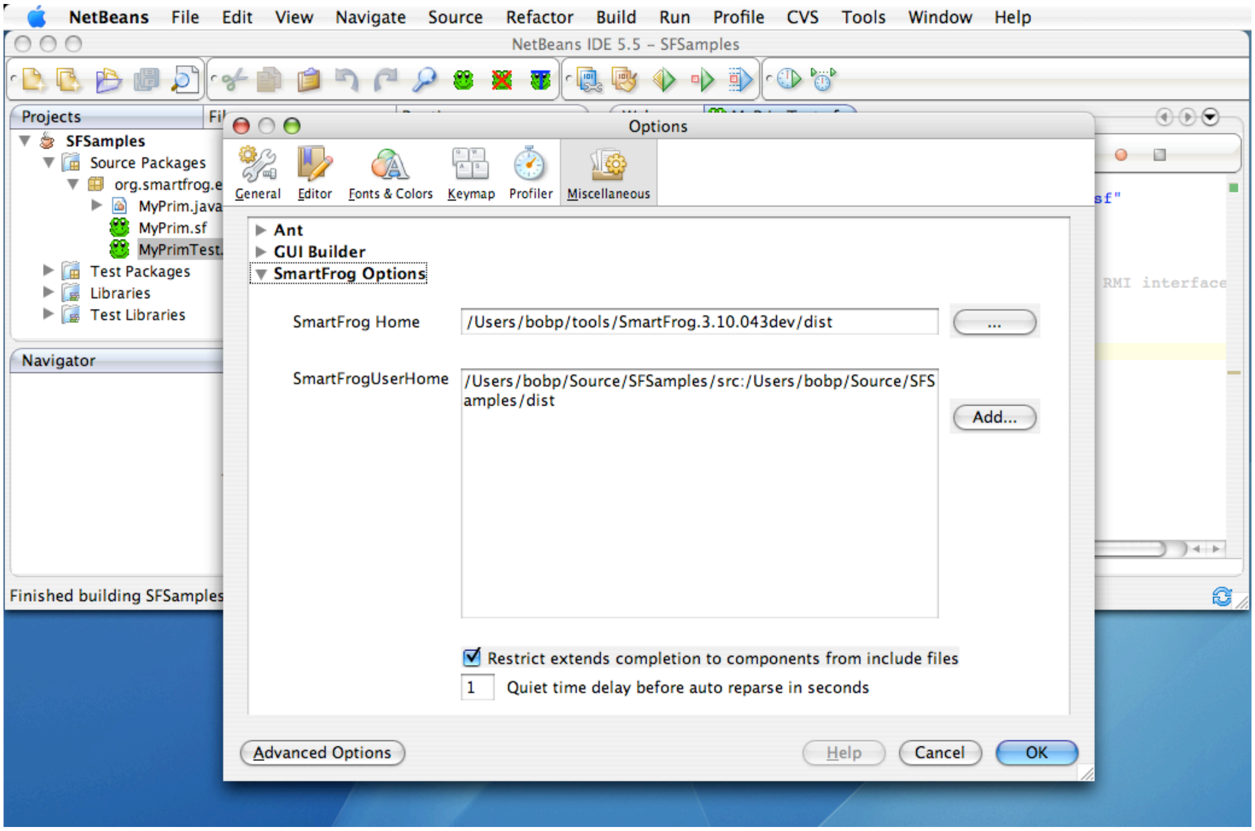


```
#include "org/smartf          .sf"
#include "org/smartf          utorial/MyPrim.sf"
#include "org/smartfrog/

sfConfig extends Compound {

MyPrimTest extends MyPrim {
    // uncomment sfExport if the component has no RMI interface
    // sfExport false;
    }
}
```

4**) Line based error indications**. A red SmartFrog icon is displayed in the editor on the line where the parser detects an error. The red icon may be displayed at the end of the file if a specific line is not identified from the exception. The user can also hover over the error and display the full SmartFrog parser error message. The previous figure shows an error.

Configuration

The plugin also provides a configuration mechanism through NetBeans preference (options) settings in the Miscellaneous section as depicted in the following figure:



SmartFrog Home /Users/bobp/tools/SmartFrog.3.10.043dev/dist

SmartFrogUserHome /Users/bobp/Source/SFSamples/src:/Users/bobp/Source/SFSamples/dist

☑ Restrict extends completion to components from include files

1 Quiet time delay before auto reparse in seconds

1) A developer is required to configure the SmartFrog Home distribution directory. This is the folder from which the SmartFrog daemon will be launched as well as the SmartFrog parser. New versions of SmartFrog can be plugged in easily using this mechanism.

2) A developer is required to configure the SmartFrog UserHome for all projects they are working on. In most cases this will point at the distribution folder where jar files are created for each project as well as the src folder for the project. Some errors may be detected if a package (jar file) has not yet been built for the project.

3) The user can configure the extends completion to show components available for all components in the SmartFrog jar files referenced in the SmartFrog Home and SmartFrogUserHome OR the user can configure the extends completion to only show the components from the include files presently specified in the description file. This is the “Restrict extends completion to components from include files” checkbox.

4) The user can also configure a delay in seconds before auto reparse is invoked. If the user has been modifying a file in the editor, the editor will wait the specified seconds before invoking a parse operation.

Installation

Three nbm files must be loaded into netbeans. These are:
org-smartfrog.nbm, org-apache-log4j.nbm, and org-smartfrog-nbm.nbm

org-smartfrog.nbm provides a wrapper for the SmartFrog jar file to be used within NetBeans. The currently wrapped library is 3.10.043.

org-apache-log4j.nbm provides a wrapper for Log4J to be used within NetBeans. The currently wrapped library is 1.2.14.

org-smartfrog-nbm.nbm provides the actual plugin.

Use the standard NetBeans techniques to manually load each of these plugins. Presently these nbm files do not contain a license string nor are they signed with a certificate.

After the nbm files are loaded, use the options panel to configure the SmartFrog distribution folder and the SmartFrog User Homes (component project locations).

Work in progress

This plugin is definitely work in progress. It was originally created to facilitate learning of both SmartFrog and NetBeans APIs. Hence structure, coding conventions, methodology can be improved in most areas. This is really a “prototype”. However, it can provide needed developer productivity when developing SmartFrog components.

Bob Potterveld