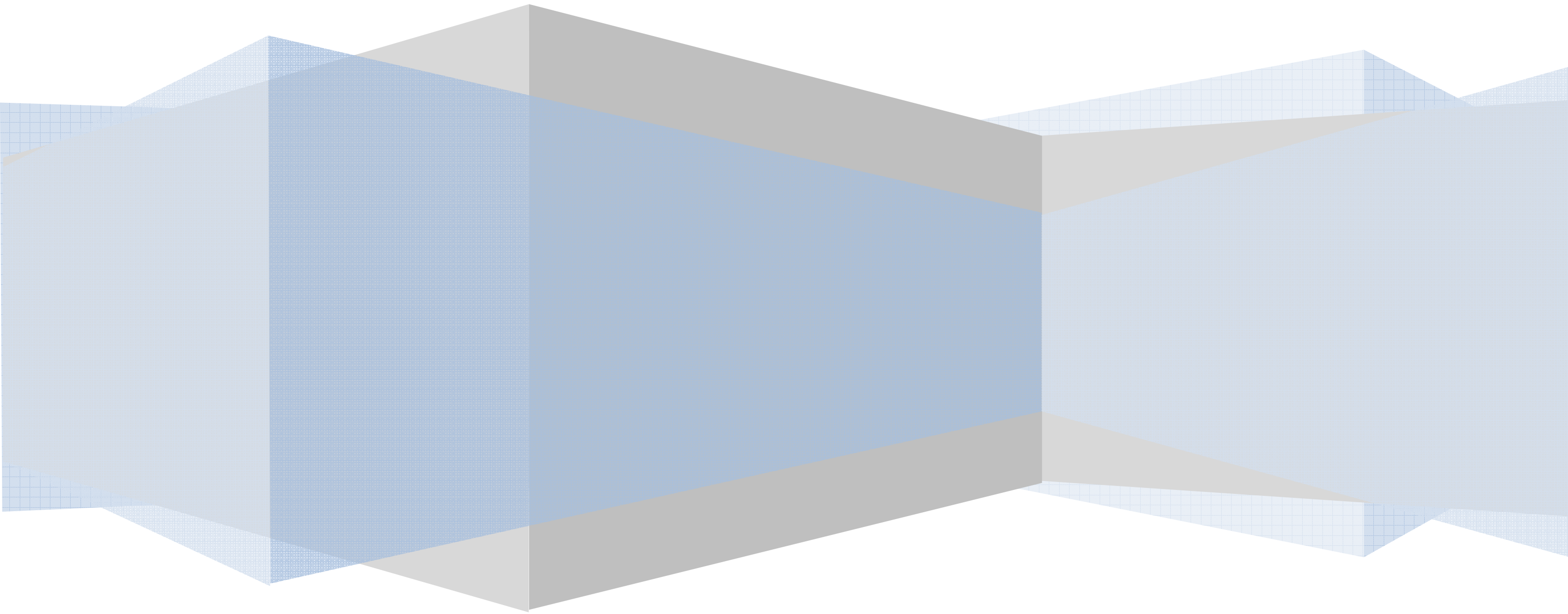
Perl On NetBeans

User Guide for version 0.2 (Beta)



Contents

Scope

“**Perl on NetBeans**” is intended for Perl programmers who want the intuitiveness of a great editor with the ease of being able to execute the program without having to do it through a command Interpreter. Also, it has the capability to inspect the austerity of your Perl programs, and format the same in accordance to the best practices and conventions.

Features of Beta 2 release

1. Perl Editor
    1. Perl file support
    2. Perl module support
    3. Syntax highlighting
    4. Brace matching
    5. Export code as HTML
2. Perl Project
    1. Create project with new sources
    2. Create project with existing sources
    3. Open existing project
    4. Add new file to project
    5. Add existing file to project
3. Code Execution
    1. Code execution
    2. Code execution with command line arguments
4. Syntax checking
5. Source code formatting (using Perl::Tidy)
6. Source code analysis (using Perl::Critic)
7. Integrated Perl-Doc
8. Configuration using the Options Panel
    1. Templates
    2. General Options
    3. Perl Tidy
    4. Perl Critic
9. Integrated versioning system (SVN/Git/Hg)

Pre-requisites

This IDE requires the following to be installed on your system for the IDE to work properly:

1. Active-Perl v5.16 (or Perl::Critic and Perl-Doc installed with your existing Perl installation)
2. Java 6

Supported platforms

1. Windows XP/7/8
2. Linux
3. Mac OS

Installation

The installation of “Perl OnNetBeans” can be summarized in the following steps:

1. Downloading the IDE (URL in the web-links section below)
2. Extracting and run the binary for your platform

The curtain raiser

Once the project is downloaded and extracted, you need to go to the folder **perlonnetbeans/bin** and use the **perl_on_netbeans** executable. The start-up screen is as shown below:

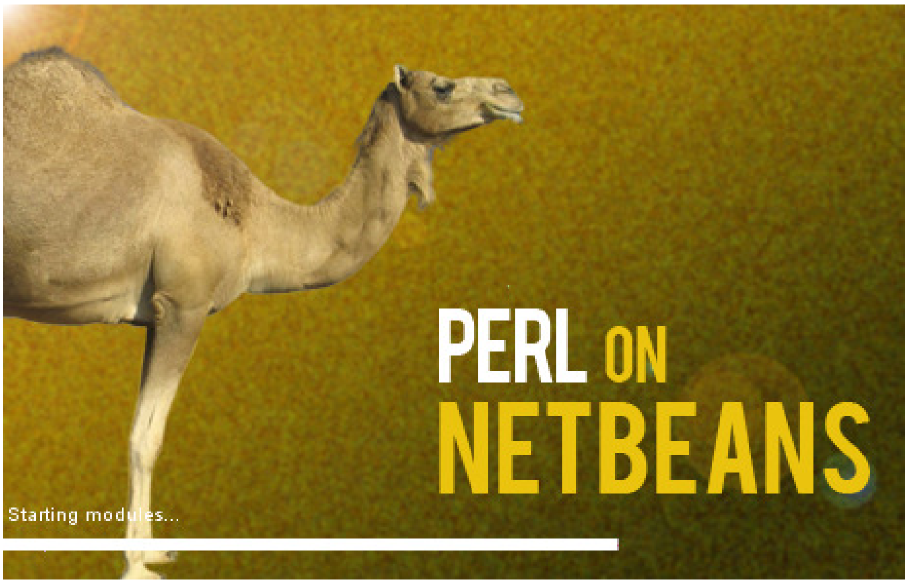


Figure 1 : The splash screen

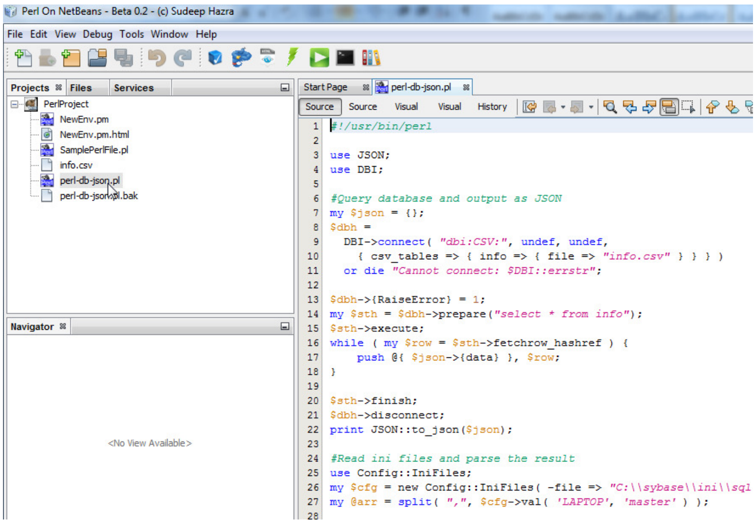
Features

The IDE currently just has the following features.

1. The Editor

1. Perl file support

Any Perl file can be opened in the editor and all available features would be available.

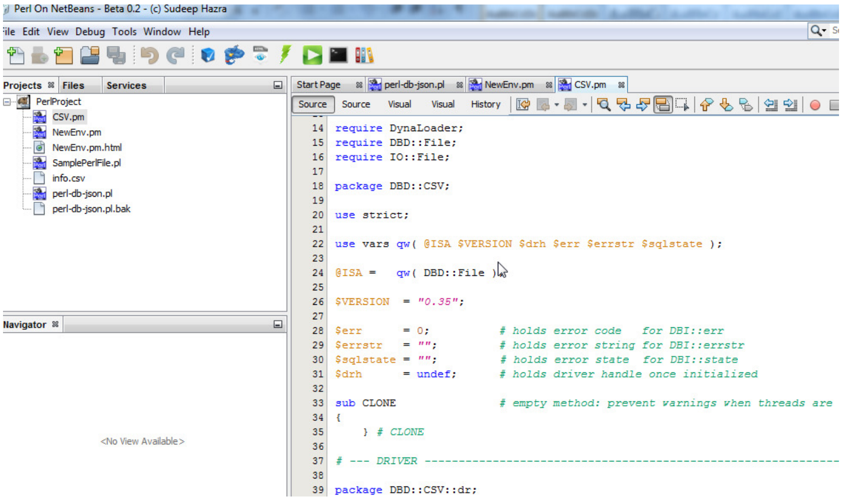


```
#!/usr/bin/perl

use JSON;
use DBI;

#Query database and output as JSON
my $json = {};
$dbh =
  DBI->connect( "dbi:CSV:", undef, undef,
    { csv_tables => { info => { file => "info.csv" } } } )
  or die "Cannot connect: $DBI::errstr";

$dbh->{RaiseError} = 1;
my $sth = $dbh->prepare("select * from info");
$sth->execute;
while ( my $row = $sth->fetchrow_hashref ) {
    push @{ $json->{data} }, $row;
}

$sth->finish;
$dbh->disconnect;
print JSON::to_json($json);

#Read ini files and parse the result
use Config::IniFiles;
my $cfg = new Config::IniFiles( -file => "C:\\sybase\\ini\\sql
my @arr = split( ",", $cfg->val( 'LAPTOP', 'master' ) );
```

Figure 2 - A Perl (*.pl) file in the editor

2. Perl module support

Any Perl module file can be opened in the editor and all available features would be available.

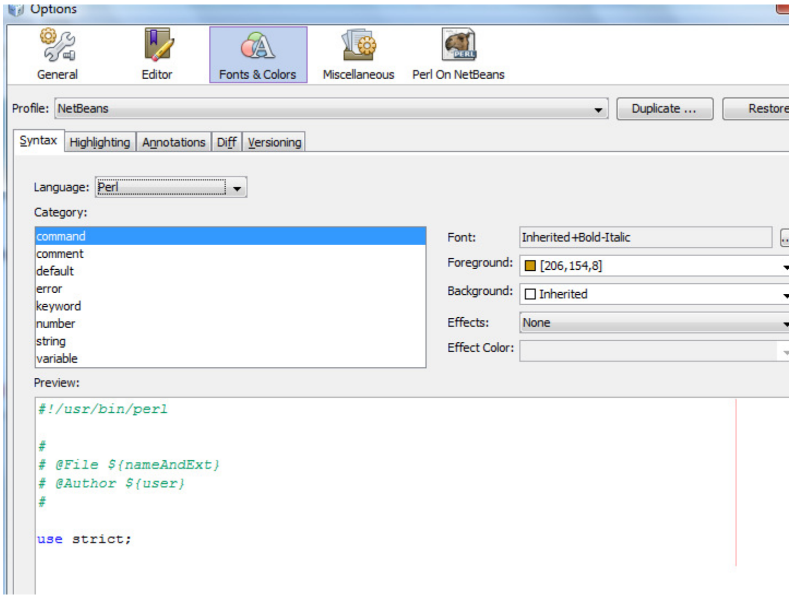


```
require DynaLoader;
require DBD::File;
require IO::File;

package DBD::CSV;

use strict;

use vars qw( @ISA $VERSION $drh $err $errstr $sqlstate );

@ISA =    qw( DBD::File )

$VERSION  = "0.35";

$err      = 0;           # holds error code   for DBI::err
$errstr   = "";          # holds error string for DBI::errstr
$sqlstate = "";          # holds error state  for DBI::state
$drh      = undef;       # holds driver handle once initialized

sub CLONE                # empty method: prevent warnings when threads are
{
    } # CLONE

# --- DRIVER -------------------------------------------------------------

package DBD::CSV::dr;
```

Figure 3 - A Perl module (*.pm) file in the editor

3. Syntax highlighting

As shown above, Perl files and modules are displayed with syntax highlighting in the editor. You also get the option of customizing the highlighted colors to your preference. Go to Tools -> Options -> Fonts and Colors. From the dropdown, select Perl and then customize as much as you want.



Profile: NetBeans

Language: Perl

Category:

- command
- comment
- default
- error
- keyword
- number
- string
- variable

Font: Inherited+Bold-Italic

Foreground: [206,154,8]

Background: □ Inherited

Effects: None

Effect Color:

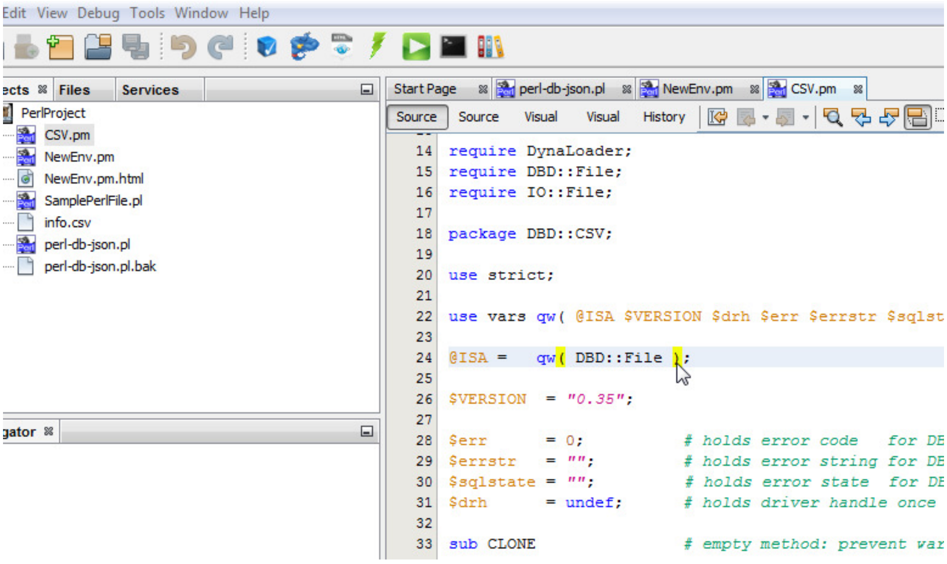
Preview:

```
#!/usr/bin/perl

#
# @File ${nameAndExt}
# @Author ${user}
#

use strict;
```

Figure 4 - Syntax highlighting options from the Options panel

4. Brace matching

This feature lets you understand the brace completion. If you create a brace mismatch, it would be highlighted in "RED" in the editor. Else, clicking on a brace will highlight the corresponding "opening/closing" brace.

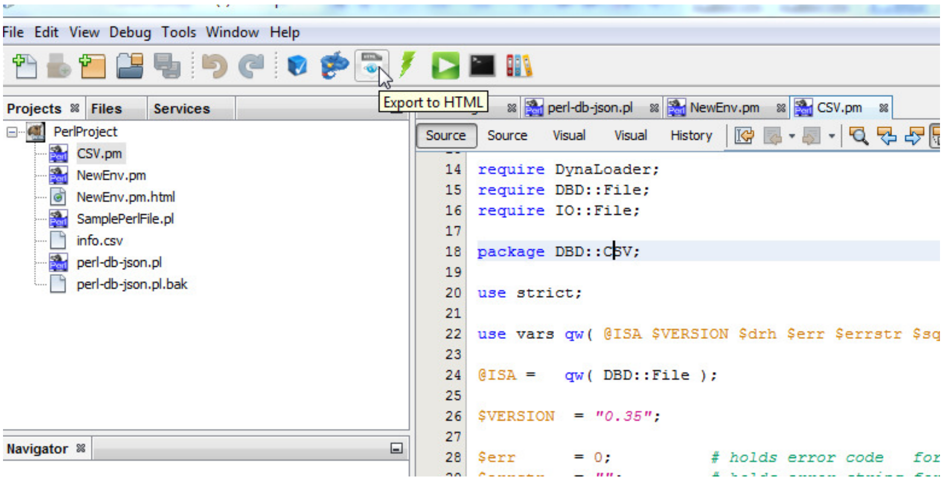


```
require DynaLoader;
require DBD::File;
require IO::File;

package DBD::CSV;

use strict;

use vars qw( @ISA $VERSION $drh $err $errstr $sqlst

@ISA =    qw( DBD::File );

$VERSION  = "0.35";

$err       = 0;            # holds error code   for DE
$errstr    = "";           # holds error string for DE
$sqlstate  = "";           # holds error state  for DE
$drh       = undef;        # holds driver handle once

sub CLONE                  # empty method: prevent war
```

Figure 5 - Brace matching in action

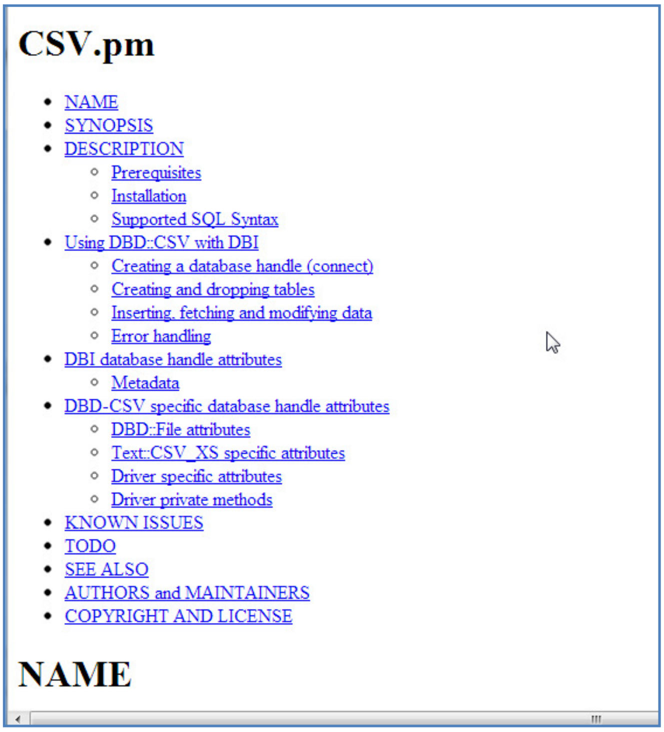
5. Export code as HTML

You can export Perl code to HTML format, using the bundled Perl::Tidy module. To do that, just open any file in the editor and click on the generate HTML button. The output HTML can be configured from the Options panel.



```
require DynaLoader;
require DBD::File;
require IO::File;

package DBD::CSV;

use strict;

use vars qw( @ISA $VERSION $drh $err $errstr $sq

@ISA =    qw( DBD::File );

$VERSION  = "0.35";

$err      = 0;          # holds error code   for
```

Figure 6 - The toolbar option to export code as HTML



CSV.pm

- NAME
- SYNOPSIS
- DESCRIPTION
  - Prerequisites
  - Installation
  - Supported SQL Syntax
- Using DBD::CSV with DBI
  - Creating a database handle (connect)
  - Creating and dropping tables
  - Inserting, fetching and modifying data
  - Error handling
- DBI database handle attributes
  - Metadata
- DBD-CSV specific database handle attributes
  - DBD::File attributes
  - Text::CSV_XS specific attributes
  - Driver specific attributes
  - Driver private methods
- KNOWN ISSUES
- TODO
- SEE ALSO
- AUTHORS and MAINTAINERS
- COPYRIGHT AND LICENSE

NAME

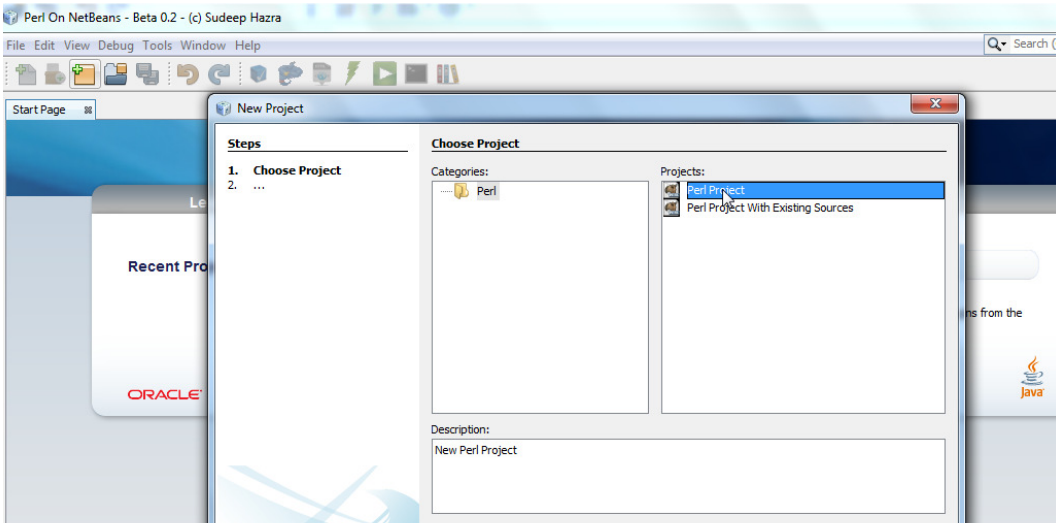
Figure 7 - The exported HTML

2. Perl Project

A Perl project can be defined as a collection of a set of Perl files that can be logically grouped together. The basic operations on a Perl Project have been outlined below.

1. Create new project

To create a new project, click on the “**New Project**” icon on the toolbar or select “**File -> New project**”. The following screen is displayed:


Perl On NetBeans - Beta 0.2 - (c) Sudeep Hazra
File Edit View Debug Tools Window Help
Search
Start Page
Recent Pro
ORACLE
Java
New Project

Steps
1. Choose Project
2. ...

Choose Project

Categories: Perl

Projects: Perl Project

Perl Project With Existing Sources

Description: New Perl Project

Figure 8 : Creating New PERL Project

Select “**Perl**” in the category and “**Perl Project**” under “**Projects**”. The subsequent screen will ask for the location where you want the project to be saved and will create a folder by that name with the project sources. The IDE will also open the project for further activities.

2. Create project with existing sources

Assuming a scenario where you have a few files that you want to create a project with and then use them. Here you can create a project with existing files using the newly available project type. Please note that all the selected files are imported to the project location and all modification would be done to the copy inside the project location.

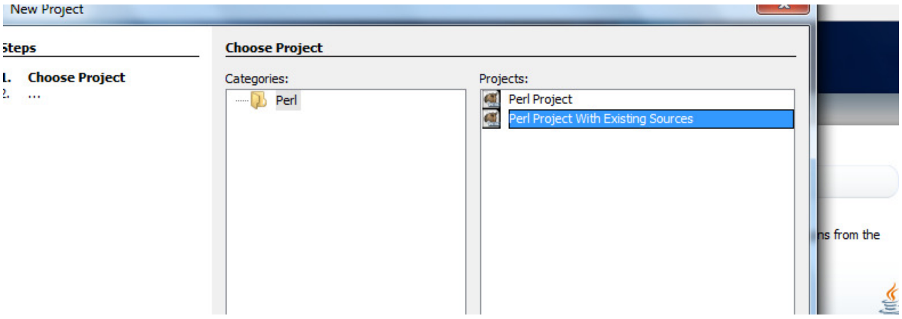
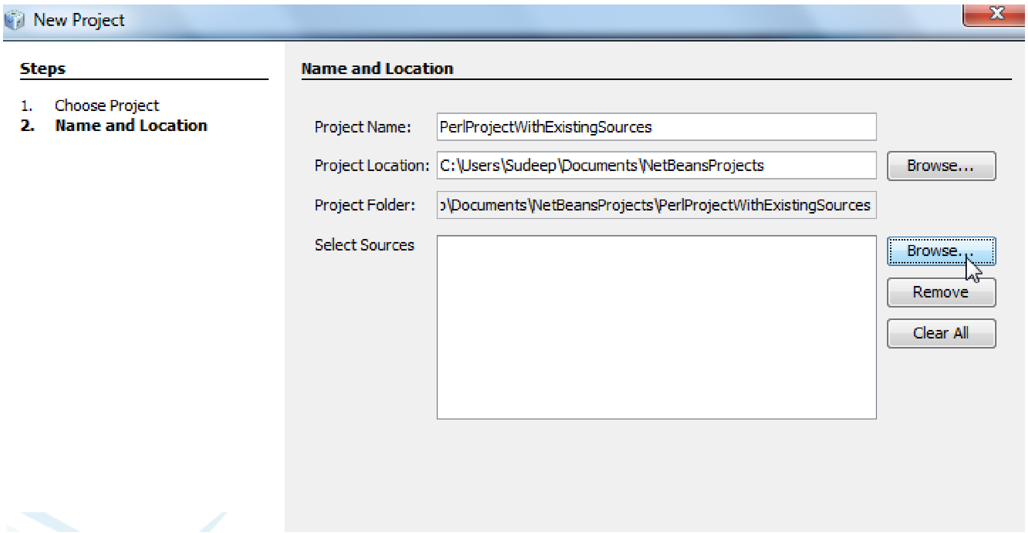


Figure 9 - Creating a project with exiting sources



Project Name: PerlProjectWithExistingSources

Project Location: C:\Users\Sudeep\Documents\NetBeansProjects

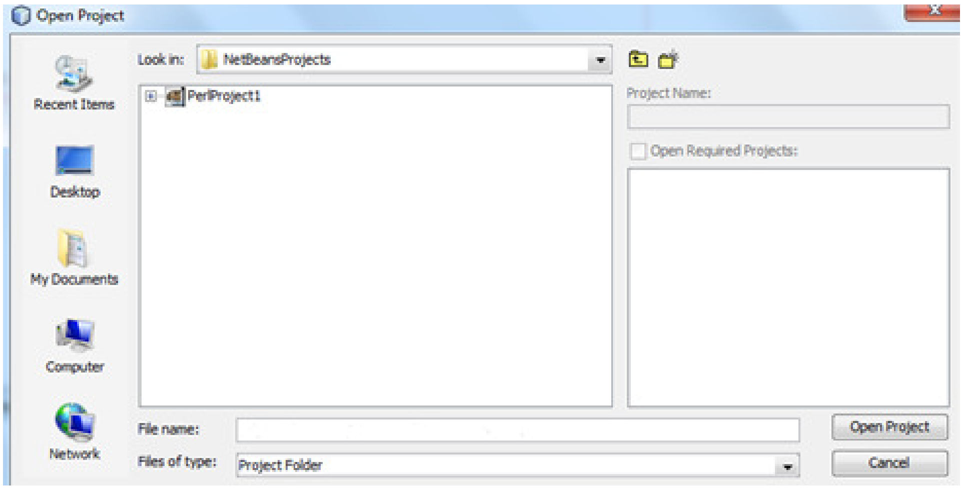
Project Folder: ɔ\Documents\NetBeansProjects\PerlProjectWithExistingSources

Select Sources

Figure 10 - Selecting the files for the project

3. Open Project

To open an already existing project, select the “**Open Project**” icon in the toolbar or go to “**File -> Open Project**”. This opens up the project selection dialog box and enable you to select am existing Perl Project to be opened. Perl project have been assigned a different icon to distinguish them from other kinds of projects, as shown below.



Look in: NetBeansProjects

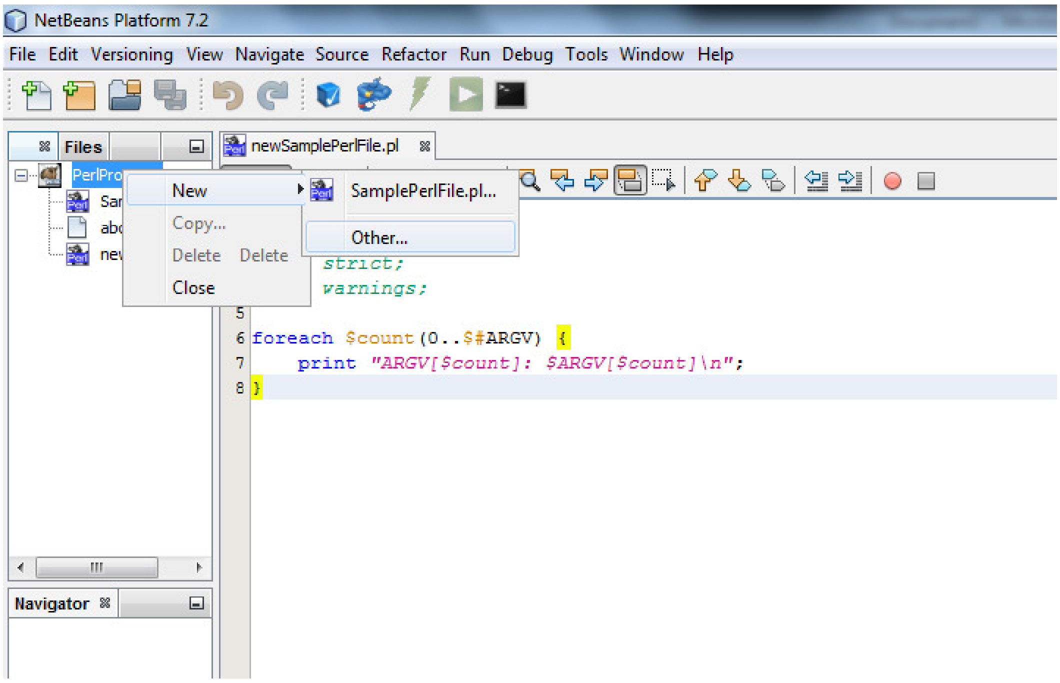
File name:

Files of type: Project Folder

Figure 11 : Opening an existing PERL Project

4. Add new file to project

A new file can be added to any project by Right Click on the project node and then by selecting “**New ->Other**” as shown below.



```
strict;
warnings;

foreach $count (0..$#ARGV) {
    print "ARGV[$count]: $ARGV[$count]\n";
}
```

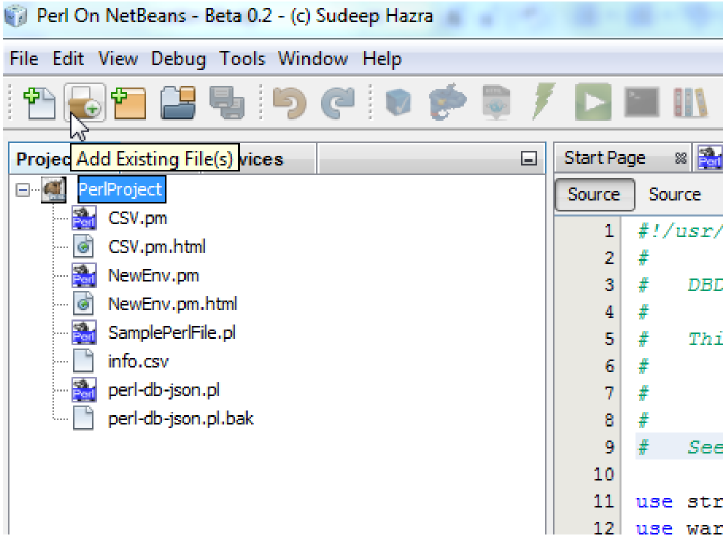
Figure 12 : Adding a new PERL file to project

This opens-up the subsequent window for selecting the file type and locations. Select the “Perl” category and then select “SamplePerlFile.pl” as the file type.

The subsequent windows will ask for the location where the files should be saved. This should be ideally left on the IDE to decide as it selects the code folder automatically.

5. Add existing file

To add an existing file to a project, select the project node and then click on the “Add existing file” icon as shown below. This will present you with another dialog window where you can select the files. Once the files are selected, they would be copied over to your project and be available under the project for all purposes.



```
#!/usr/
#
#    DBD
#
#    Thi
#
#
#
#    See

use str
use war
```

Figure 13 : Toolbar icon to add existing file to a project

3. Check Files for Syntax Errors

This feature enables you to check the syntax of the file that is currently selected in the editor. To execute any specific file that is currently selected in the editor, click on the “**Syntax Check**” button on the toolbar as shown below. This checking happens based on the version of the Perl compiler that is currently installed on your system.

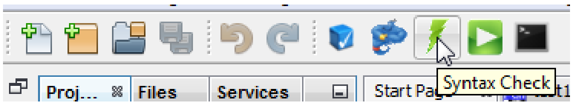
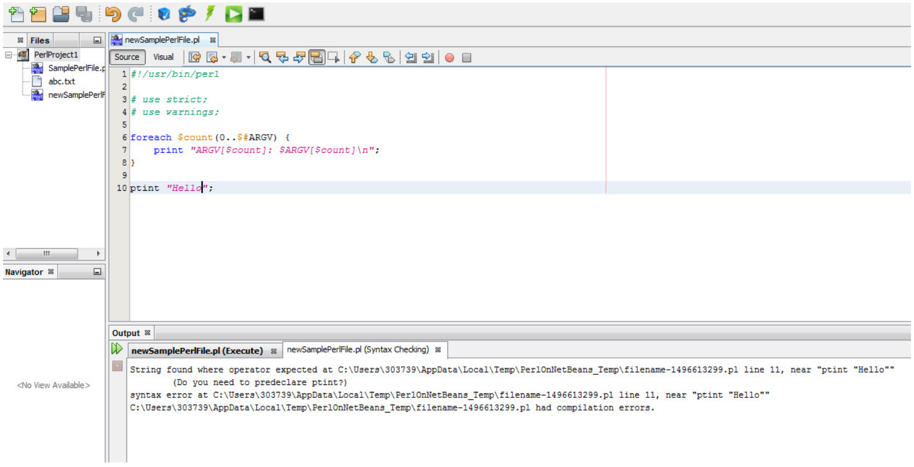


Figure 14 : Toolbar icon to check for syntax errors in selected PERL file



```
#!/usr/bin/perl

# use strict;
# use warnings;

foreach $count (0..$#ARGV) {
    print "ARGV[$count]: $ARGV[$count]\n";
}

ptint "Hello";
```

```
String found where operator expected at C:\Users\303739\AppData\Local\Temp\PerlOnNetBeans_Temp\filename-1496613299.pl line 11, near "ptint "Hello""
        (Do you need to predeclare ptint?)
syntax error at C:\Users\303739\AppData\Local\Temp\PerlOnNetBeans_Temp\filename-1496613299.pl line 11, near "ptint "Hello""
C:\Users\303739\AppData\Local\Temp\PerlOnNetBeans_Temp\filename-1496613299.pl had compilation errors.
```

Figure 15 : Syntax checking results

4. Code execution

1. Execute files

This feature enables you to execute the file that is currently selected in the editor. To execute any specific select it in the editor and click on the “**Execute**” button on the toolbar as shown below. This execution happens based on the version of the Perl compiler that is currently installed on your system or selected from the options panel (*described below*).

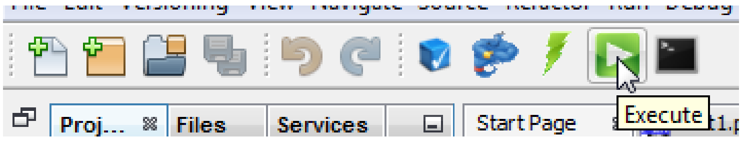


Figure 16 : Toolbar icon to execute selected PERL file

2. Code execution with command line arguments

The latest release allows one to supply command-line argument(s). To execute a file with command line options, select the file in the editor and click the “**Execute (with command line options)**” button on the toolbar as shown below. On using this feature, a dialog box pops up, in turn facilitating the input of command-line parameters to be used in one’s program.

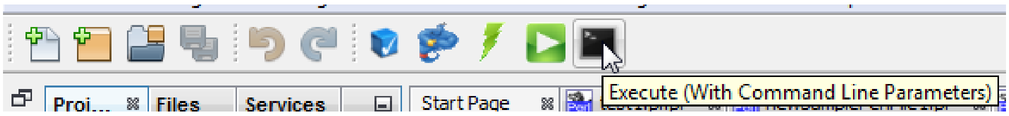
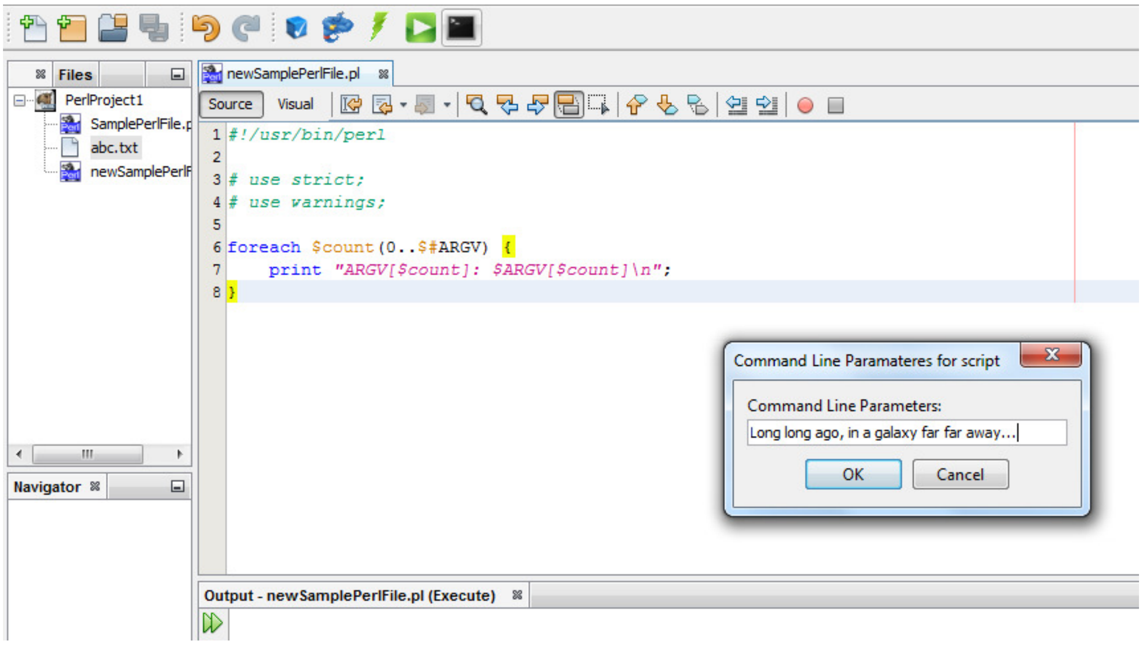


Figure 17 : Toolbar icon to execute selected PERL file with command line arguments



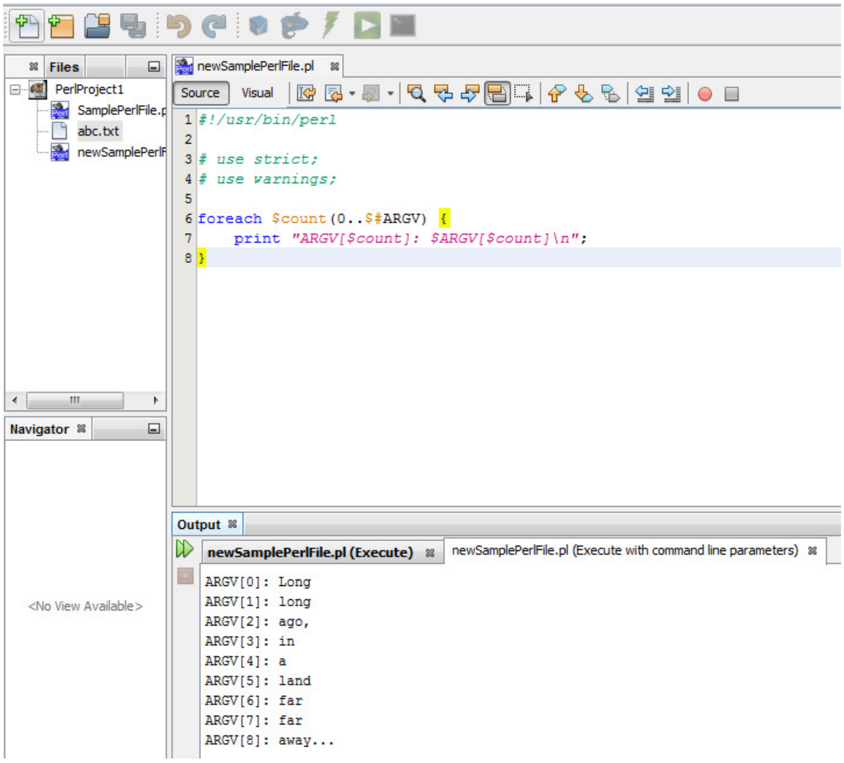
```
#!/usr/bin/perl

# use strict;
# use warnings;

foreach $count (0..$#ARGV) {
    print "ARGV[$count]: $ARGV[$count]\n";
}
```

Command Line Paramateres for script

Command Line Parameters: Long long ago, in a galaxy far far away...

Figure 18 : Providing the command line arguments

Subsequent Execution of the script



```
#!/usr/bin/perl

# use strict;
# use warnings;

foreach $count (0..$#ARGV) {
    print "ARGV[$count]: $ARGV[$count]\n";
}
```

```
ARGV[0]: Long
ARGV[1]: long
ARGV[2]: ago,
ARGV[3]: in
ARGV[4]: a
ARGV[5]: land
ARGV[6]: far
ARGV[7]: far
ARGV[8]: away...
```

Figure 19 : Results of execution with the command line arguments

5. Source Code formatting

This feature enables you to perform source re-formatting as per **Perl::Tidy**. To perform analysis, select the file in the editor and click on the code formatter button on the toolbar as shown below. This will run the source code formatter against the selected source.

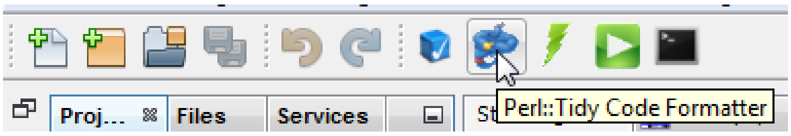
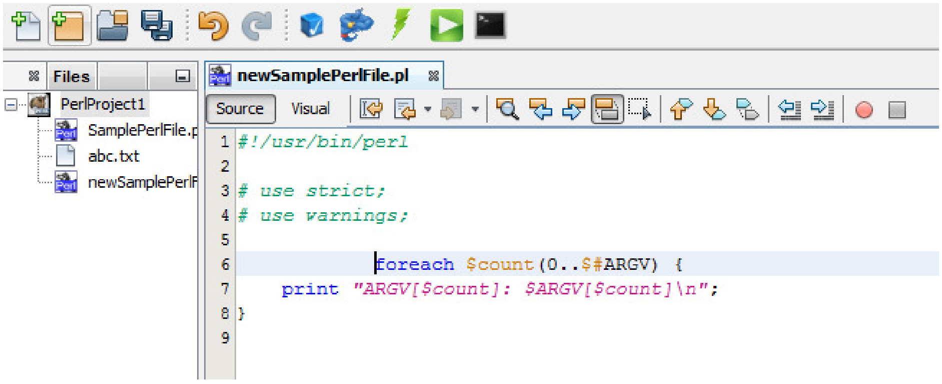
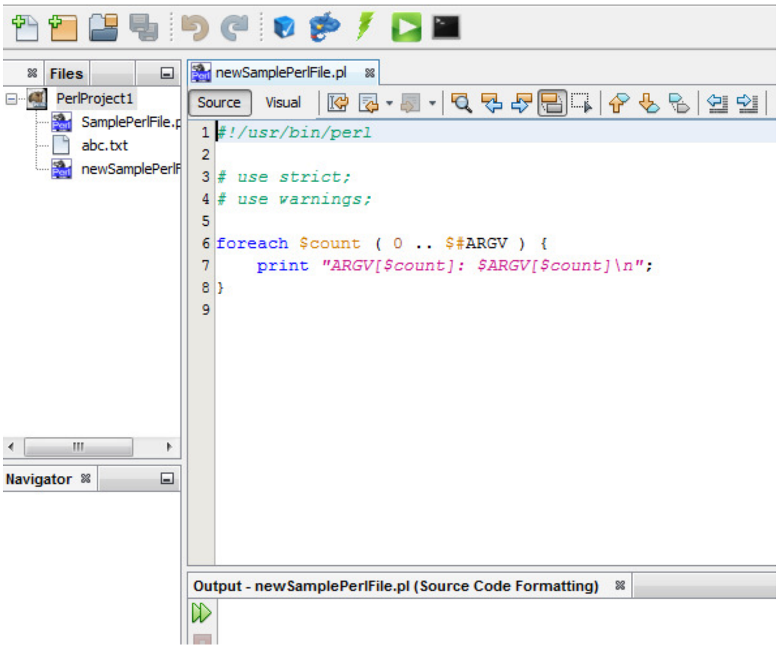


Figure 20 : Toolbar icon to format selected PERL file



```
#!/usr/bin/perl

# use strict;
# use warnings;

            foreach $count (0..$#ARGV) {
    print "ARGV[$count]: $ARGV[$count]\n";
}

```

Figure 21 : Code formatting (Before)



```
#!/usr/bin/perl

# use strict;
# use warnings;

foreach $count ( 0 .. $#ARGV ) {
    print "ARGV[$count]: $ARGV[$count]\n";
}

```

Figure 22 : Code Formatting (After)

6. Source code analysis

This feature enables you to perform source analysis as per **Perl::Critic**. To perform analysis, select the file in the editor and click on the code analyzer button on the toolbar as shown below. This will run the source code analyzer against the selected source. You'll need **Perl::Critic** installed.

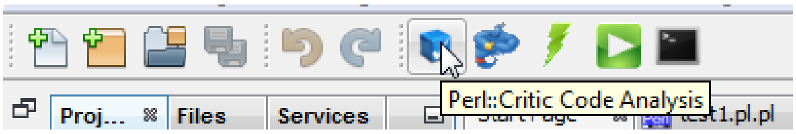
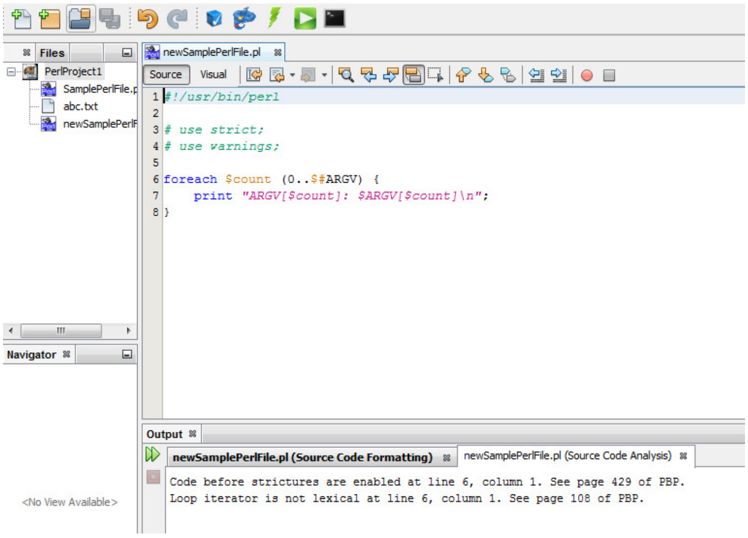


Figure 23 : Toolbar icon to analyze source code of selected PERL file

The output of the analysis will be provided in the Output window, as shown in the screenshot below:



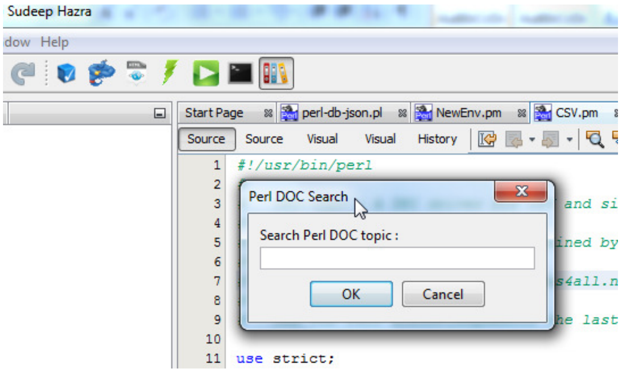
```
1 #!/usr/bin/perl
2
3 # use strict;
4 # use warnings;
5
6 foreach $count (0..$#ARGV) {
7     print "ARGV[$count]: $ARGV[$count]\n";
8 }
```

```
Code before strictures are enabled at line 6, column 1. See page 429 of PBP.
Loop iterator is not lexical at line 6, column 1. See page 108 of PBP.
```

Figure 24 : Source code analysis results

7. Integrated PerlDOC support

PerlDOC support has been also made available. If you want more information about any keyword, just select it in your editor and press on the "PerlDOC" button, and it will display the documentation in the output window. Similarly, if you just press the button, a pop-up window will ask you for the help topic and display it in the output window. You'll need **perl-doc** installed for this.



```
#!/usr/bin/perl
use strict;
```

Perl DOC Search

Search Perl DOC topic :

OK Cancel

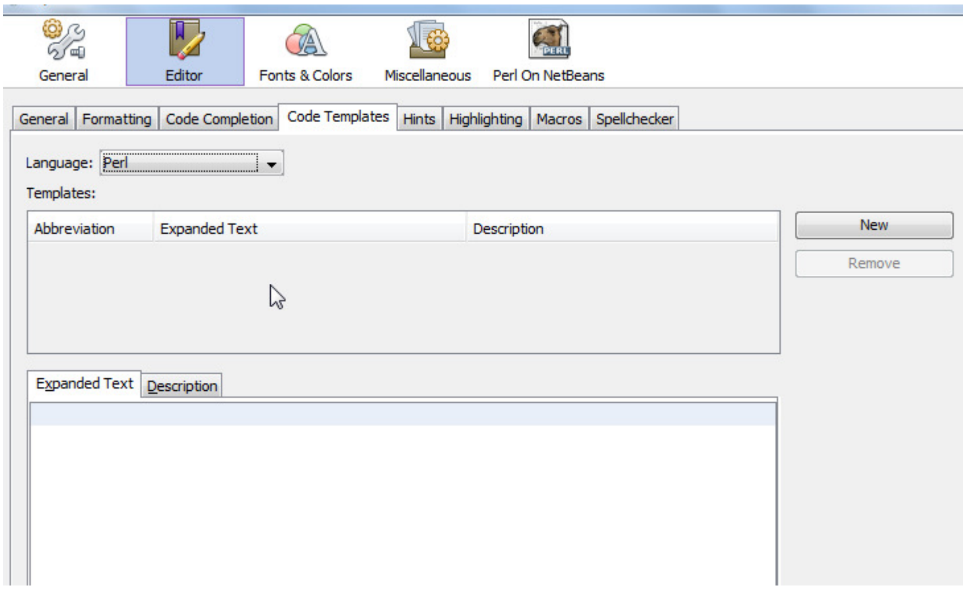
Figure 25 - Toolbar icon to display documentation

8. Options Panel

The Options panel contains a section called "Perl OnNetBeans" for our usage. This tab contains 3 sub-tabs that are utilized for various options. Also template support has been added for programming efficiency.

1. Templates

You can specify any code template that you use regularly as a shortcut here. For example "p" can translate to "print". So, if you add a template here, and then go to your editor with a Perl file open, type "p" and press "tab", and voila, this would get translated in the editor as "print". You can add as many templates as you want.



Language: Perl

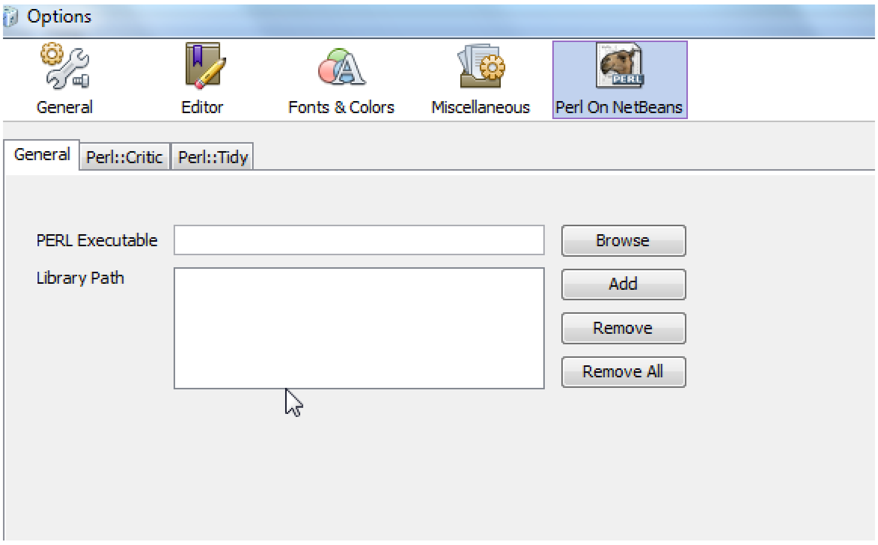
Templates:

Abbreviation	Expanded Text	Description

Figure 26 : Code template configuration panel

2. General

This tab contains the option to over-ride the default PERL installed on your machine. Let's say that you have PERL 5.14 installed on your machine as the default but you want to use PERL 5.16 available at a non-default location, you can point the perl.exe binary using this tab and your program will be operated on by the executable specified here. If you do not specify anything here, the default PERL will be used. You can also specify any custom library file location here. These locations would be used as an addition to your @INC in the script.



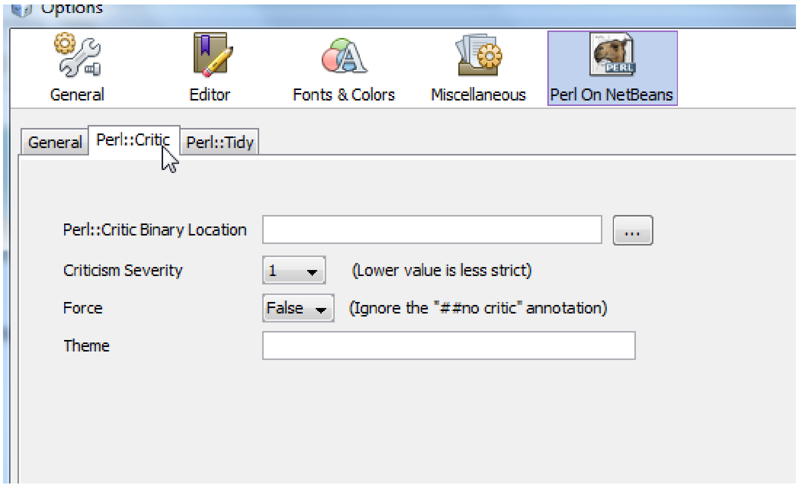
PERL Executable

Library Path

Figure 27 : PERL binary configuration panel

3. Perl::Critic

This tab can be used to customize the code analysis options. For a complete meaning of the options here, please refer to the URL http://linux.die.net/man/1/perlcritic.



Perl::Critic Binary Location

Criticism Severity 1 (Lower value is less strict)

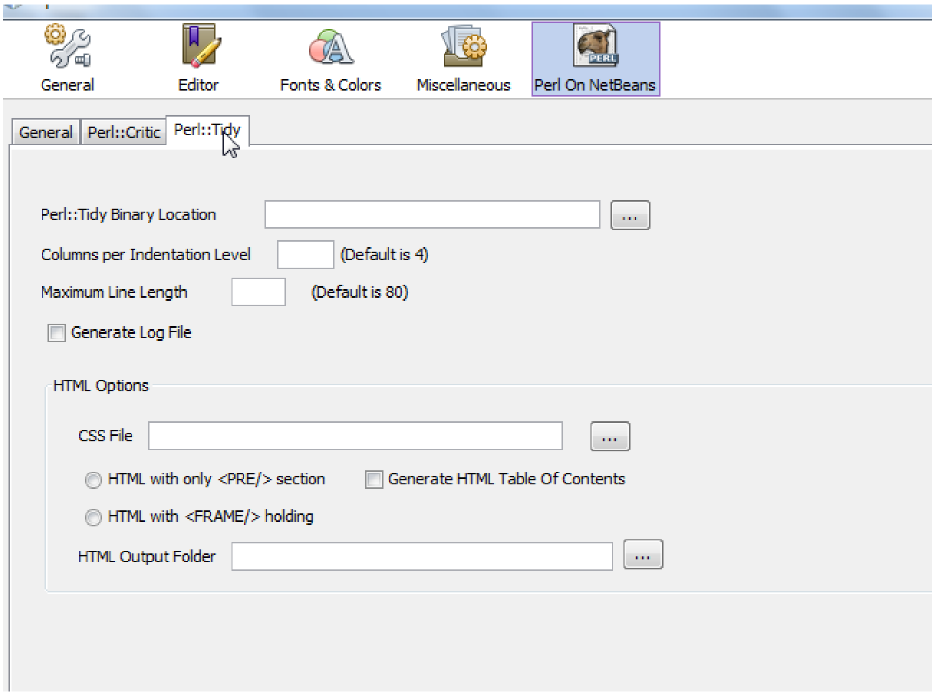
Force False (Ignore the "##no critic" annotation)

Theme

Figure 28 : PERL::Critic (Code Analyzer) configuration panel

4. Perl::Tidy

This tab can be used to customize the code formatting options. For a complete meaning of the options here, please refer to the URL http://perltidy.sourceforge.net/perltidy.html.



Perl::Tidy Binary Location [ ] ...

Columns per Indentation Level [ ] (Default is 4)

Maximum Line Length [ ] (Default is 80)

☐ Generate Log File

HTML Options

CSS File [ ] ...

○ HTML with only <PRE/> section ☐ Generate HTML Table Of Contents

○ HTML with <FRAME/> holding

HTML Output Folder [ ] ...

Figure 29 : PERL::Tidy (Code Formatting) configuration panel

We encountered a feature of "Export to HTML" and it can be configured from this panel. You can specify the output folder of the generated HTML (project folder by default), specify if you require a "Table Of Contents", specify any CSS file, and also FRAME and PRE tags in the output HTML.

9. Integrated versioning system

The new version also includes integrated support for major code versioning systems including Subversion, Git and Mercurial. The appropriate context menu options are displayed based on whether the code is connected with a SVN/Git/Hg repository.

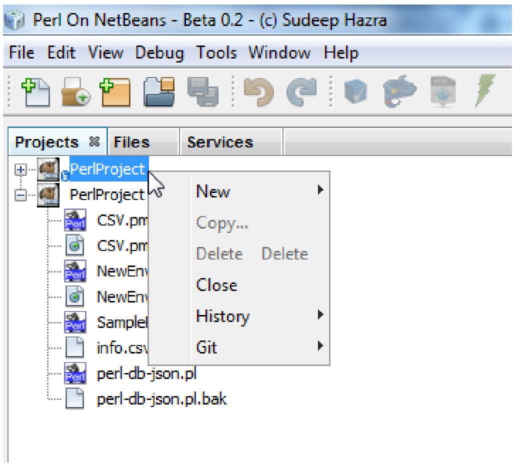


Figure 30 : A Git repo Project

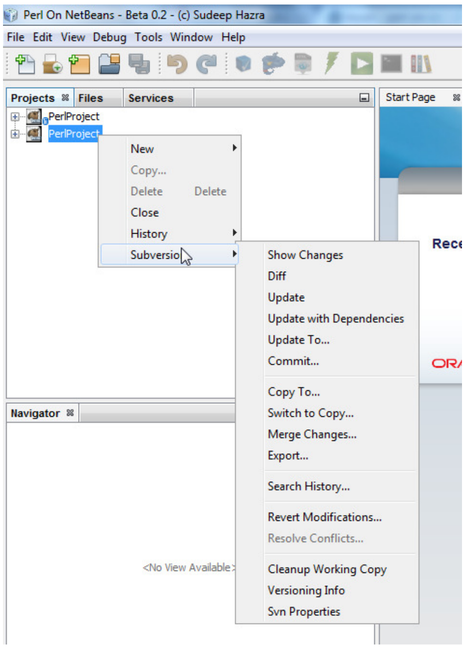
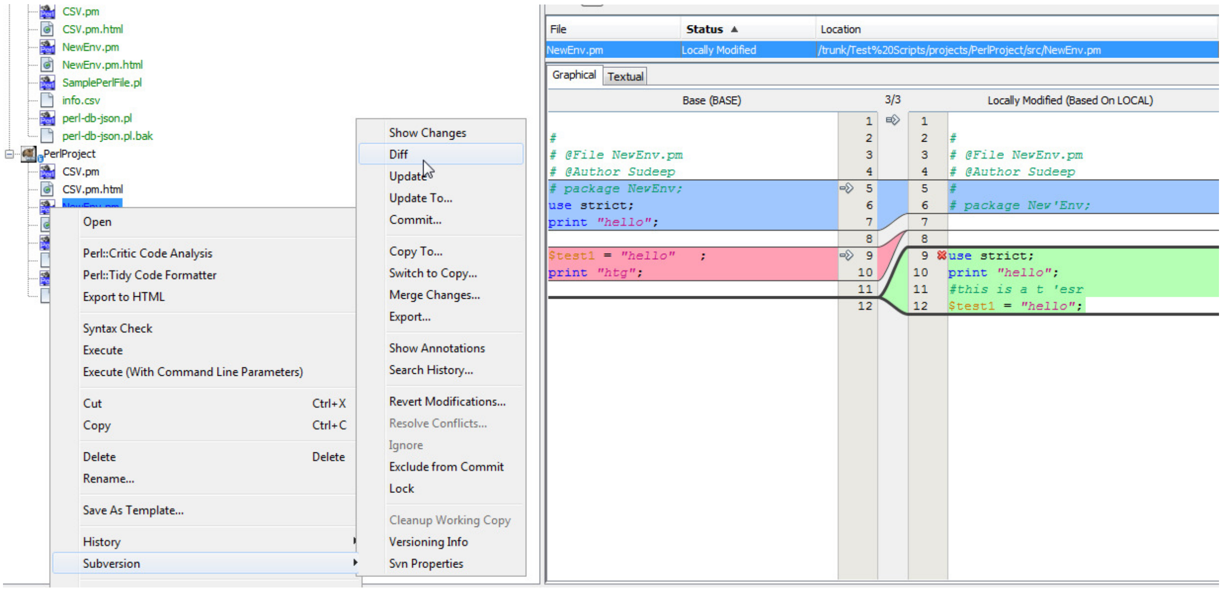


Figure 31 : A SVN repo Project



File	Status ▲	Location
NewEnv.pm	Locally Modified	/trunk/Test%20Scripts/projects/PerlProject/src/NewEnv.pm

```
#
# @File NevEnv.pm
# @Author Sudeep
# package NevEnv;
use strict;
print "hello";

$test1 = "hello"    ;
print "htg";
```

```
#
# @File NevEnv.pm
# @Author Sudeep
#
# package Nev'Env;


use strict;
print "hello";
#this is a t 'esr
$test1 = "hello";
```

Figure 32 : A Subversion file diff

Feedback

Since this is a community driven initiative, I would request everyone to log-on to my Google Code space for downloading the IDE. Also for any issues, please do log an issue with Google Code on the project space. And for any comments/suggestions, please do send a mail to the Google Groups group-id as mentioned in the “Web Links” section below.

Web Links

1. The IDE download page – http://code.google.com/p/perl-on-netbeans
2. FaceBook page - http://www.facebook.com/PerlOnNetBeans
3. Perl On NetBeans user group – http://groups.google.com/group/perlonnetbeans
4. Google Groups - perlonnetbeans@googlegroups.com
5. Active-Perl download - http://www.activestate.com/activeperl/downloads
6. Java download - http://www.oracle.com/technetwork/java/javase/downloads/index.html

Credits

1. Anupam Chowdhury for the new and refreshing splash screen.
2. Asad Shahabuddin for the multi-platform tests on Mac OS.