Easy Cover and EUnit

Testing with Erllde

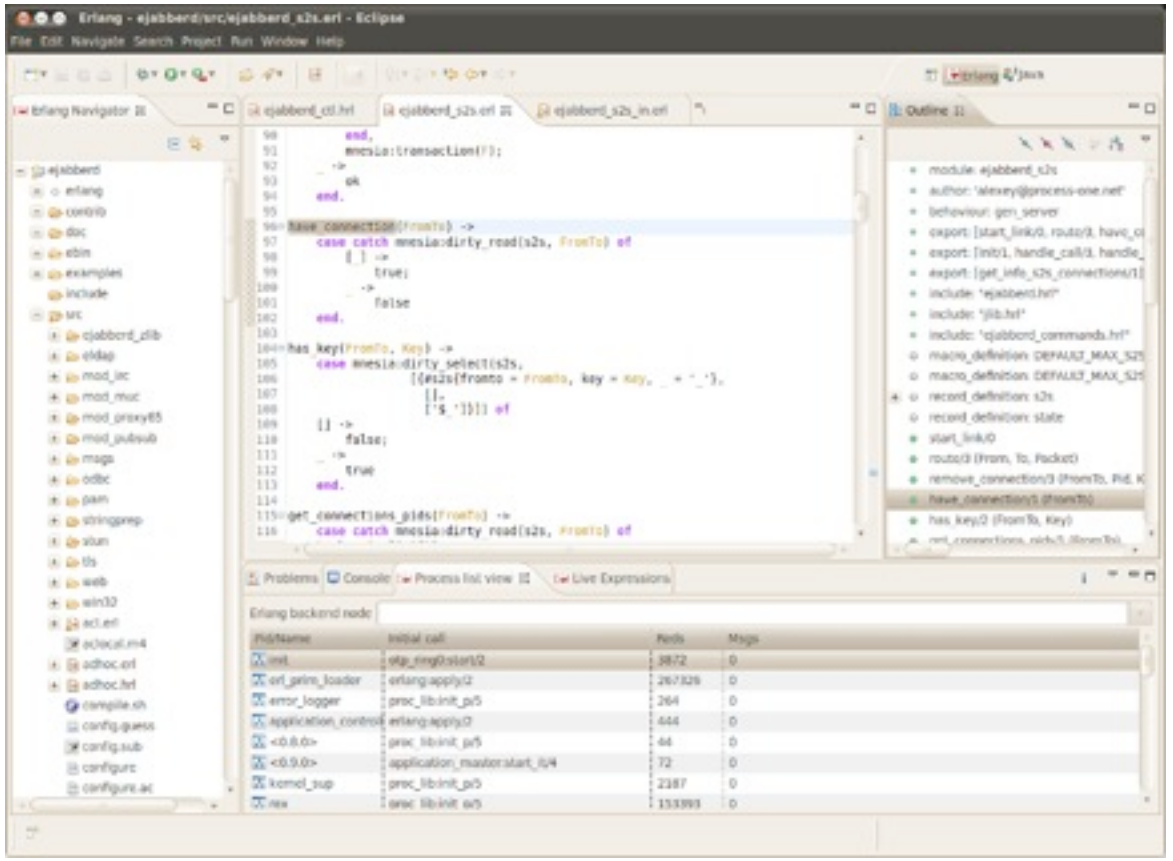
Aleksandra Lipiec
aleksandra.lipiec@erlang-solutions.com

Outline

- About ErlIde
- EUnit and Cover fundamentals
- Purpose of creating Cover Plugin
- Features
- Let's try it!
- Perspectives
- Overview of other ErlIde plugins

About ErlIde

An Eclipse plugin for Erlang development

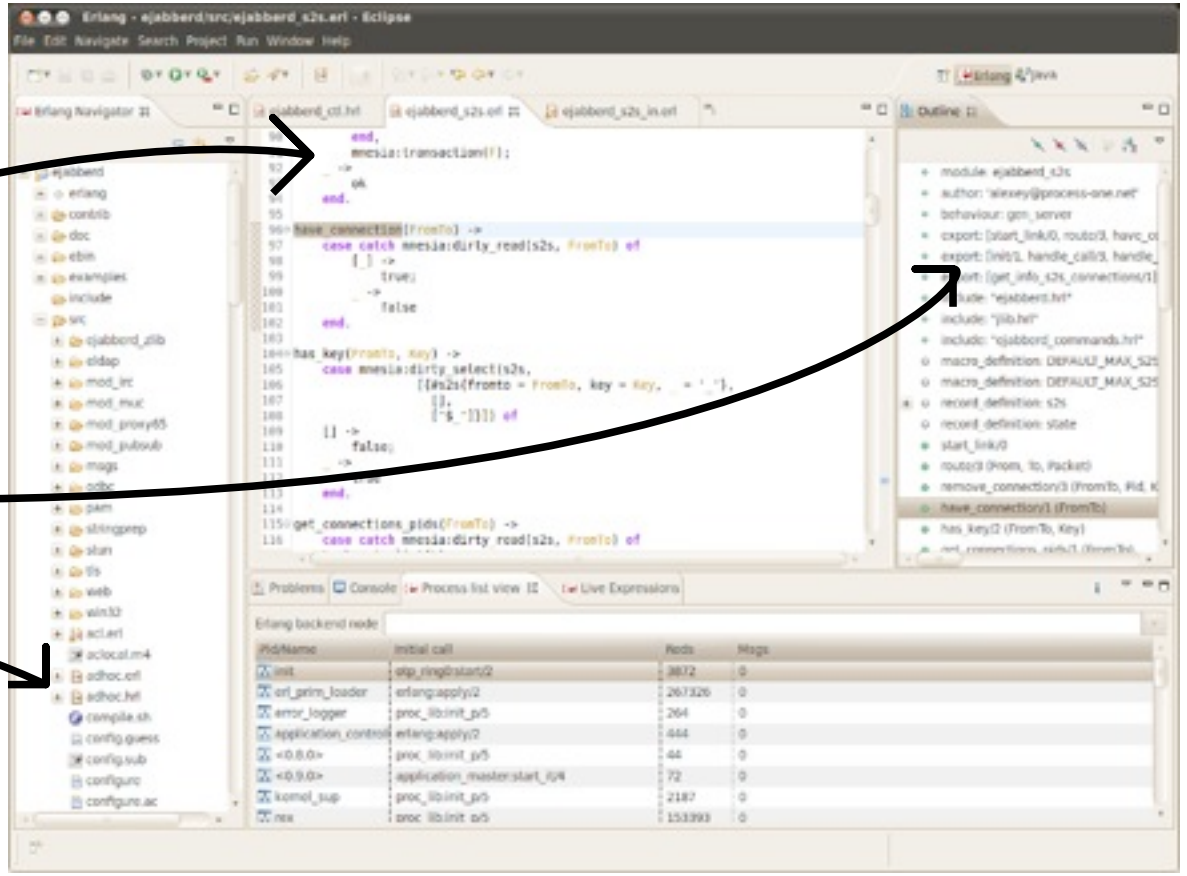


Project's website: http://erlide.org/index.html

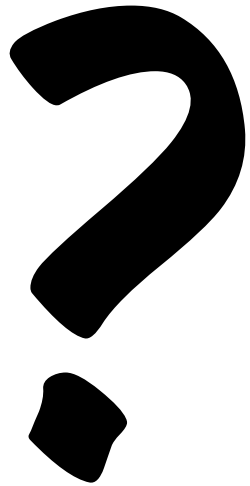
About ErlIde

Main features:

- Erlang editor
- Integrated compiler
- Navigation
- Project support
- Integrated debugger



EUnit and Cover fundamentals



EUnit – what is it?

- A unit testing framework for Erlang
- Inspired by JUnit, adapted to functional and concurrent programming
- Uses macros
- A unit can be a function, a **module**, a process or a whole application

EUnit – how to use it?

- Modules need to include the EUnit library: `-include_lib("eunit/include/eunit.hrl").`
- Test functions can be in a module `mod` or `mod_tests`
- Test function names should match `..._test()` (simple tests) or `..._test_()` (tests generators)
- Run the tests with `eunit:test(mod)` or `mod:test()`

EUnit example

```
-module(eunitex).
-include_lib("eunit/include/eunit.hrl").

reverse([]) ->
    [];
reverse([H|T]) ->
    [reverse(T)|H].

reverse_nil_test() ->
    ?assertEqual([], reverse([])).
reverse_one_test() ->
    ?assertEqual([1], reverse([1])).
reverse_many_test() ->
    ?assertEqual([3,2,1], reverse([1,2,3])).
```

EUnit example

```
eunitex: reverse_one_test...*failed*
::error:{assertEqual_failed,[{module,eunitex},
                             {line,12},
                             {expression,"reverse ( [ 1 ] )"},
                             {expected,[1]},
                             {value,[[]|1]}]}
  in function eunitex:'-reverse_one_test/0-fun-0-'/1


eunitex: reverse_many_test...*failed*
::error:{assertEqual_failed,[{module,eunitex},
                             {line,14},
                             {expression,"reverse ( [ 1 , 2 , 3 ] )"},
                             {expected,[1,2,3]},
                             {value,[[[[]|3]|2]|1]}]}
  in function eunitex:'-reverse_many_test/0-fun-0-'/1


=======================================================
  Failed: 2.  Skipped: 0.  Passed: 1.
error
```

EUnit example

```
-module(eunitex).
-include_lib("eunit/include/eunit.hrl").

reverse([]) ->
    [];
reverse([H|T]) ->
    reverse(T) ++ [H].

reverse_nil_test() ->
    ?assertEqual([], reverse([])).
reverse_one_test() ->
    ?assertEqual([1], reverse([1])).
reverse_many_test() ->
    ?assertEqual([3,2,1], reverse([1,2,3])).
```

```
  All 3 tests passed.
ok
```

Cover - what is it?

- A coverage analysis tool for Erlang
- Counts how many times each executable line of code is executed when the program is run
- Requires modules to be **cover compiled**
- Information about the calls is kept in an internal database
- Can be run on multiple nodes

Cover - how to use it?

- Prepare module **`mod`**
- Start cover **`cover:start()`**
- Cover compile module **`cover:compile_module(mod)`**
  (available also: compile_directory, compile_beam, compile_beam_directory)
- Run tests
- Perform coverage analysis **`cover:analyse(mod)`**
- Create a report **`cover:analyse_to_file(mod, "output.html", [html])`**

Cover example

```
-module(stack).
-compile(export_all).

new() -> {stack, []}.

push(El, {stack, S}) when length(S) < 1000 ->
    {stack, [El | S]};
push(_El, {stack, _S}) ->
    erlang:error('out of mem').

pop({stack, []}) ->
    erlang:error('empty stack');
pop({stack, [H | T]}) ->
    {H, {stack, T}}.

empty({stack, []}) -> true;
empty({stack, _}) -> false.

size({stack, S}) -> length(S).
```

Cover example

- Example results

```
{ok,[{{stack,new,0},{1,0}},
     {{stack,push,2},{1,1}},
     {{stack,pop,1},{1,1}},
     {{stack,empty,1},{1,1}},
     {{stack,size,1},{0,1}},
     {{stack,stack_test,0},{5,1}}]}
```

Cover example

- Example results file

```
File generated from /home/wirenth/Pulpit/stack.erl by COVER 2011-10-18 at 16:52:41

******************************************************************************

        |  -module(stack).
        |  -compile(export_all).
        |
        |  new() ->
    1..|      {stack, []}.
        |
        |  push(El, {stack, S}) when length(S) < 1000 ->
    2..|     {stack, [El | S]};
        |  push(_El, {stack, _S}) ->
    0..|      erlang:error('out of mem').
        |
        |  pop({stack, []}) ->
    0..|      erlang:error('empty stack');
        |  pop({stack, [H | T]}) ->
    1..|      {H, {stack, T}}.
        |
        |  empty({stack, []}) ->
    0..|      true;
        |  empty({stack, _}) ->
    1..|      false.
        |
        |  size({stack, S}) ->
    0..|      length(S).
        |
```

Purpose of integrating Cover and EUnit in ErlIde

- Facilitate cover (and eunit) usage
- Provide a friendly graphical interface
- Simplify testing

Features

- Presenting test results in a form of a tree
- Providing coverage statistics
  (per function, module, source folder, project)
- Marking coverage in the editor
- HTML coverage report generation
- Saving and restoring coverage results

More features

- Browsing coverage marking for specified modules and functions
- Opening items from statistics view
- Potential ability to cooperate with any testing plugin (CommonTest, QuickCheck)
- Available for Eclipse >= 3.5 (Erllde >= 0.11.1)

Let's try it!

Installation

- Install Erlang R12B-5 or later
- Install Eclipse 3.6 or later
- Install ErlIde with Cover plugin
  - Go to ***Help*** → ***Install New Software…*** → ***Available software***
  - In the dialog choose ***Add…*** and enter **http://erlide.org/update**
  - Choose **Erlang IDE** and ***Erlang code coverage*** from ***Erlang IDE add-ins (optional)***
- Set Erlang runtime
  - Go to ***Window*** → ***Preferences*** → ***Erlang*** → ***Installed runtimes***
  - Add path for your Erlang installation
- Done!

Exercise 1

Implementing simple tests for a module and performing coverage analysis

1) Import **listdb** project
2) Run coverage analysis
3) Implement more tests to achieve 100% code coverage

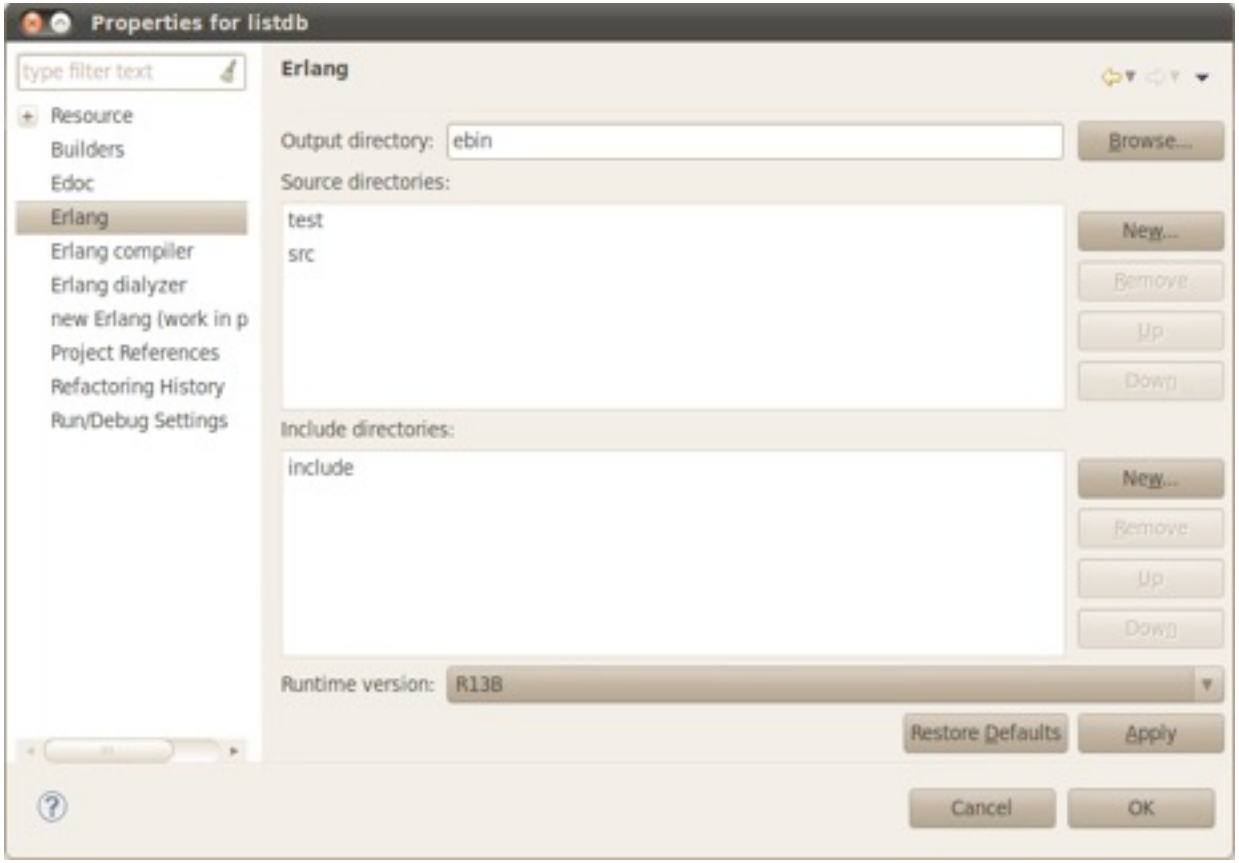
Import a project

Go to **File → Import → Erlang → Import Erlang project into workspace**

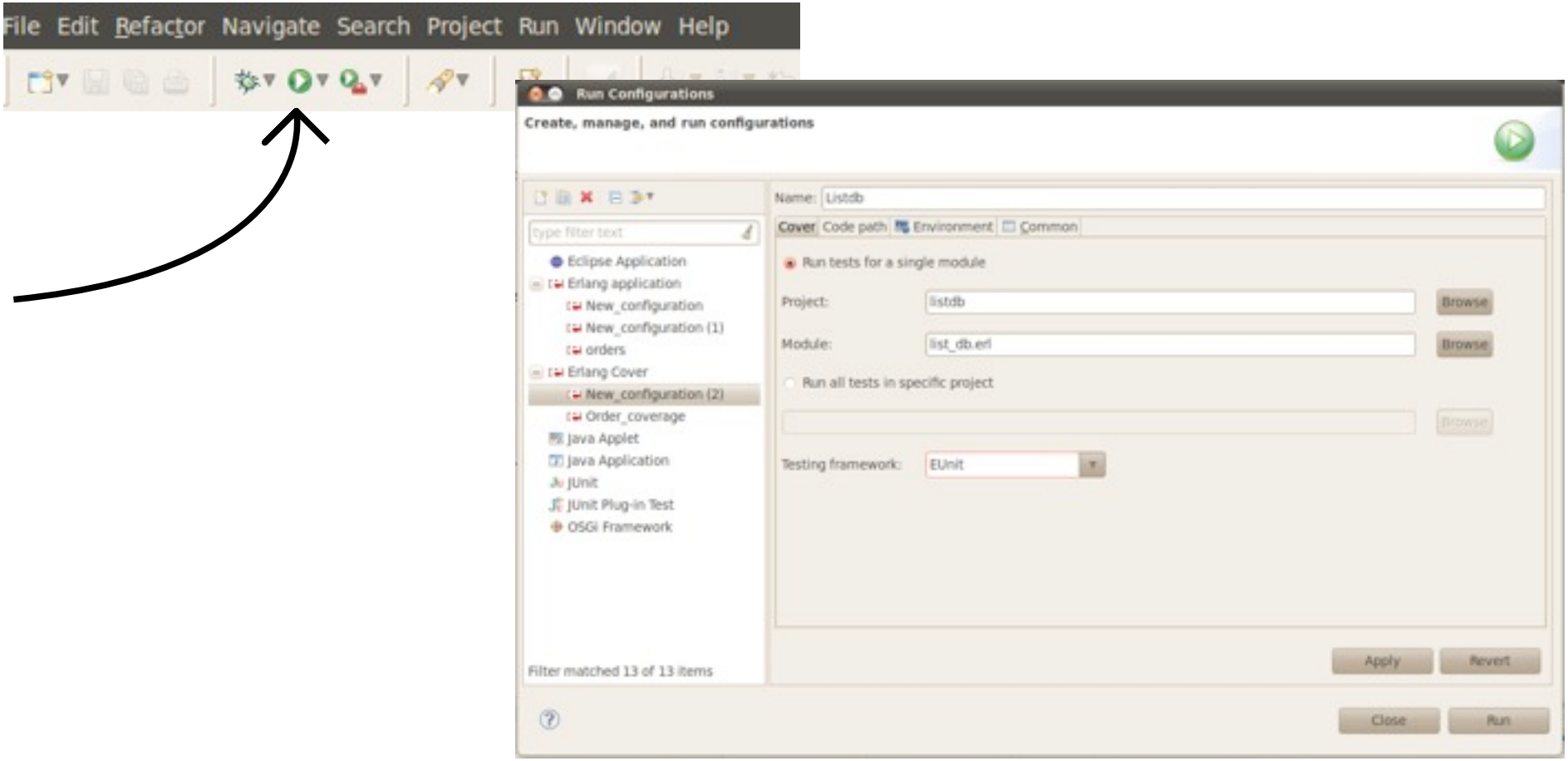
- Browse **listdb** project
- Check if **src** and **test** folders are marked as source folders
- Finish

Directory stucture

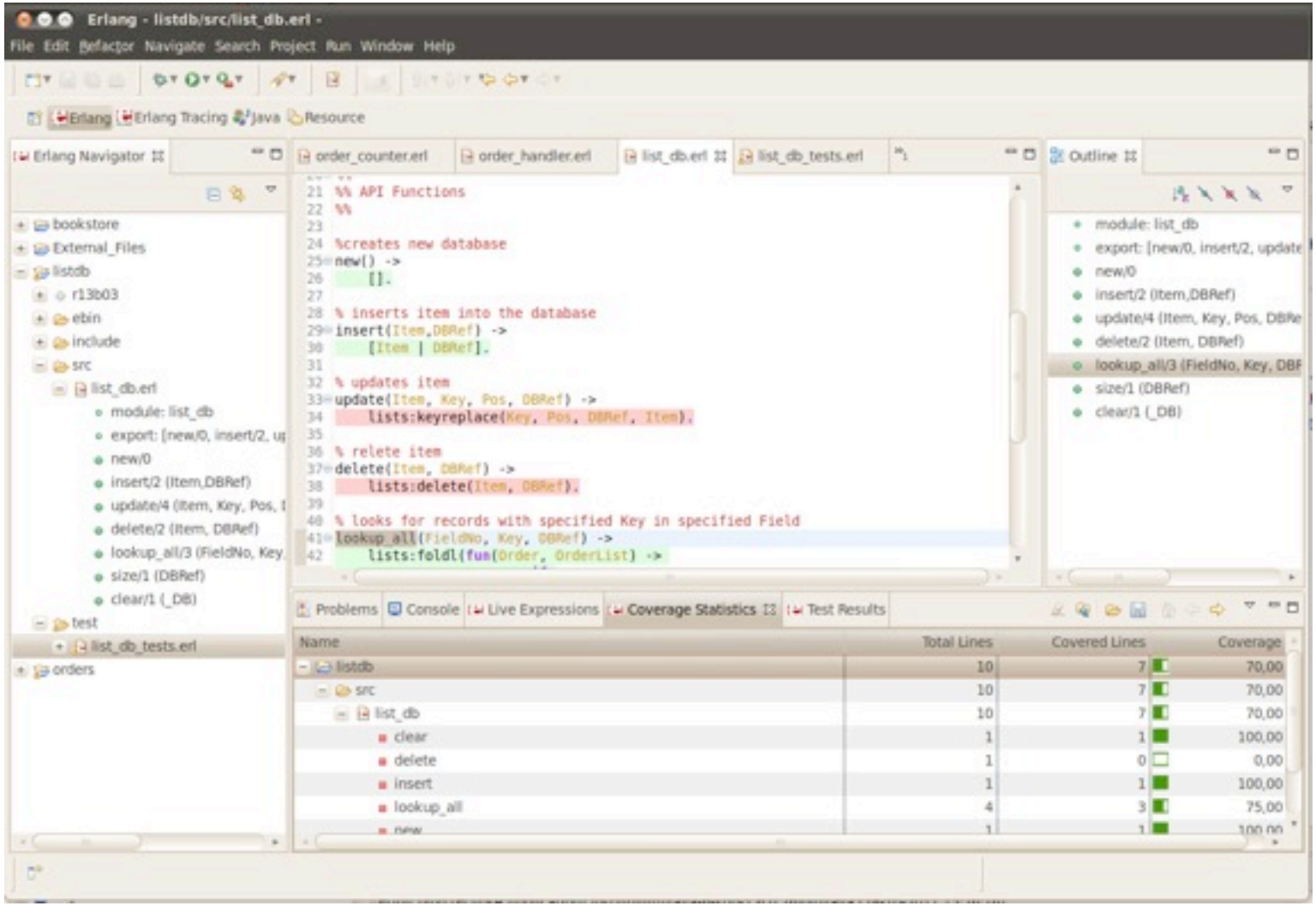
Configuration of directories with Erlang files



Run coverage analysis



Results



Exercise 2

Adding a test files to a project

1) Import **bookstore** project
2) Add test folder to the project
3) Add new test files
4) Implement tests
5) Run coverage analysis for the project

Exercise 3

Exporting results to HTML

1) Browse HTML results
2) Export results to HTML
3) See exported results

Perspectives

- Sorting statistics
- Integration with testing plugin
- Distributed code coverage analysis
- Other...

Other plugins for ErlIde

1. Wrangler plugin – a refactoring tool
2. Tracing plugin – TTB integration

Further information

About Cover Plugin:

https://github.com/erlide/erlide/wiki/Cover-plugin

About Erllde:

https://github.com/erlide/erlide/wiki

Thank you for your attendance!