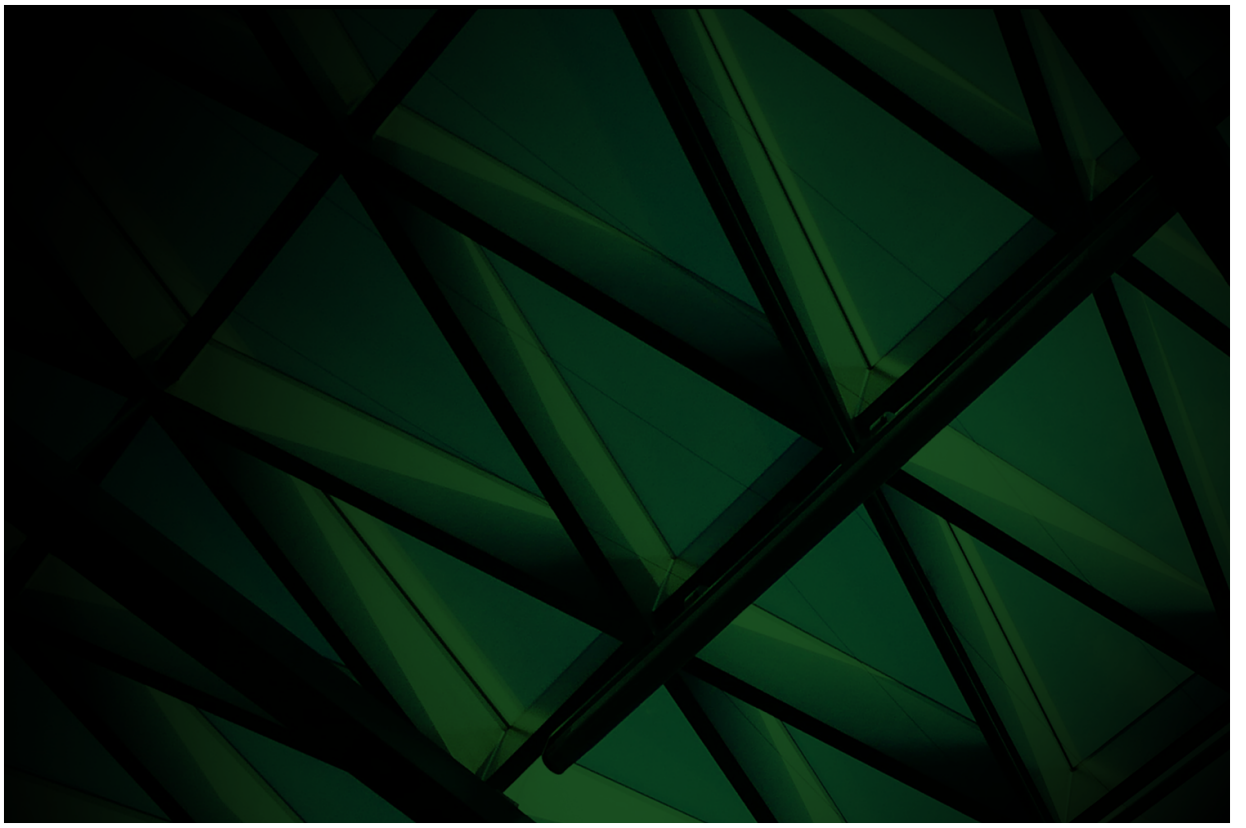


Code Coverage White Paper

VERSION 2022

jade platform

Contents

Code Coverage

Code coverage is a measure used in software engineering to describe the degree to which the system's source code has been executed. It is a useful measure to assure the quality of a set of tests, as opposed to directly reflecting the quality of the system under test.

Code coverage can help testers and developers to:

- Discover methods and blocks of code that are not exercised by a set of tests
- Create tests that increase code coverage
- Quantify the overall code coverage of a schema, which is one measure of quality

This feature provides code coverage:

- Analysis of Jade methods, indicating those lines of code that have been executed and those that have not been executed.
- Information in a two-step process. Firstly, while the code is exercised via appropriate test routines, the interpreter logs required information to a flat file. You can then examine this file by running the **CodeCoverage** application.

The analysis of code coverage information provides the following measures of coverage for a schema.

- The percentage of Jade methods in the schema that have been executed.
- The percentage of classes in the schema with Jade methods that have been included in the coverage.
- The percentage of primitive types in the schema with Jade methods that have been included in the coverage.
- The percentage of blocks in the executed methods that have been covered.

A *block* of code represents a Jade interpreter operation. For a fuller definition and examples of blocks and instructions for the collection of code coverage results, see "Code Coverage", in Chapter 17 of the *Developer's Reference*.

In addition, you can extract a Comma-Separated Values (CSV) file that contains a list of all Jade methods for the schema that have not been executed in the coverage.

The analysis of code coverage information provides the following measures of coverage for a class and primitive type.

- The percentage of Jade methods that have been executed.
- The percentage of blocks in the executed methods that have been covered.
- The number of times each method has been called.

In addition, you can view a list of the methods that have not been executed for the class or primitive type in the schema.

Note Ensure that code coverage uses one process only at a time, as running multiple applications carrying out code coverage can result in incorrect coverage percentages.

Enabling Code Coverage

You can control code coverage programmatically or via the Jade system tray menu.

- Programmatic control

  You can use the following methods defined on the **JadeProfiler** class to control code coverage for the current process.

  - **startCodeCoverage**, which starts code coverage
  - **stopCodeCoverage**, which stops code coverage
  - **reportCodeCoverage**, which reports all code coverage data
  - **resetCodeCoverage**, which clears all code coverage data

  Note Use a transient instance of the **JadeProfiler** class as the receiver.

- Jade system tray menu

  The Jade menu in the system tray provides a menu item to control code coverage for an application. This feature provides a way of enabling code coverage dynamically, without having to change application code.

  The **Start**, **Stop**, **Report**, and **Reset** menu options perform the same functionality as the programmatic methods in the previous list.

  The **View** menu option stops the code coverage session, automatically initiates the code coverage application, and displays the result of the code coverage file created.

  Note Using this initiation method, the application must be already running. Only those methods executed *after* code coverage is started are reported on.

Example of an Application Enabling Code Coverage

One way of enabling code coverage programmatically is to declare a new application that initiates a special **initialize** method, as follows.

```
initializeCodeCoverage() updating;
begin
    create myCodeCoverage;  // myCodeCoverage is a reference to JadeProfiler on
app
    myCodeCoverage.startCodeCoverage(); // start coverage
    myInitialize();                     // call standard initialize
end;
```

In the standard **finalize** method, turn off coverage if it is on, as follows.

```
myFinalize() updating;
begin
    if myCodeCoverage <> null then
        myCodeCoverage.stopCodeCoverage();
        myCodeCoverage.reportCodeCoverage();
        delete myCodeCoverage;
    endif;
```

```
    ...
end;
```

Controlling the Code Coverage File Location

You can use the **CodeCoverageDirectory** parameter in the [JadeProfiler] section of the Jade initialization file to control the location of the code coverage files output by the interpreter.

When running as a Jade presentation client, the application server's initialization file is used to determine the directory location, and the files are written to the application server's environment.

When running as a standard Jade client, the initialization file defined by the client is used to determine the directory location, and the files are written to the client's environment.

By default, the code coverage files are written to the **logs\codeCoverage** subdirectory of the Jade program data directory. (For details, see "Program Data Directory Location", in Chapter 2 of the *Installation and Configuration Guide*.)

If the value of the **CodeCoverageDirectory** parameter in the initialization file is not an absolute path, the value is appended to the program data directory to construct the absolute path. Any directories specified are created if they do not exist.

By default, the file names have the following format.

```
application-name_yyyymmdd_hhmmss.ccd
```

You can set the file name programmatically via the **codeCoverageFileName** property of the **JadeProfiler** class. If this file name is not an absolute path name, it is appended to the effective value of the **CodeCoverageDirectory** parameter. If the property value is empty (the default), the default file name is constructed.

Analyzing Code Coverage Files

You can initiate the **CodeCoverage** application via the:

- **Code Coverage** command in the Jade Platform development environment File menu.

  This initiates the **CodeCoverage** application. The application displays a Code Coverage Results Browser, which is initially empty.

  From the File menu, select the **Load** command and then select the appropriate command from the Load submenu. These commands initiate a common File Open dialog, which enables you to select the coverage file for files to be loaded.

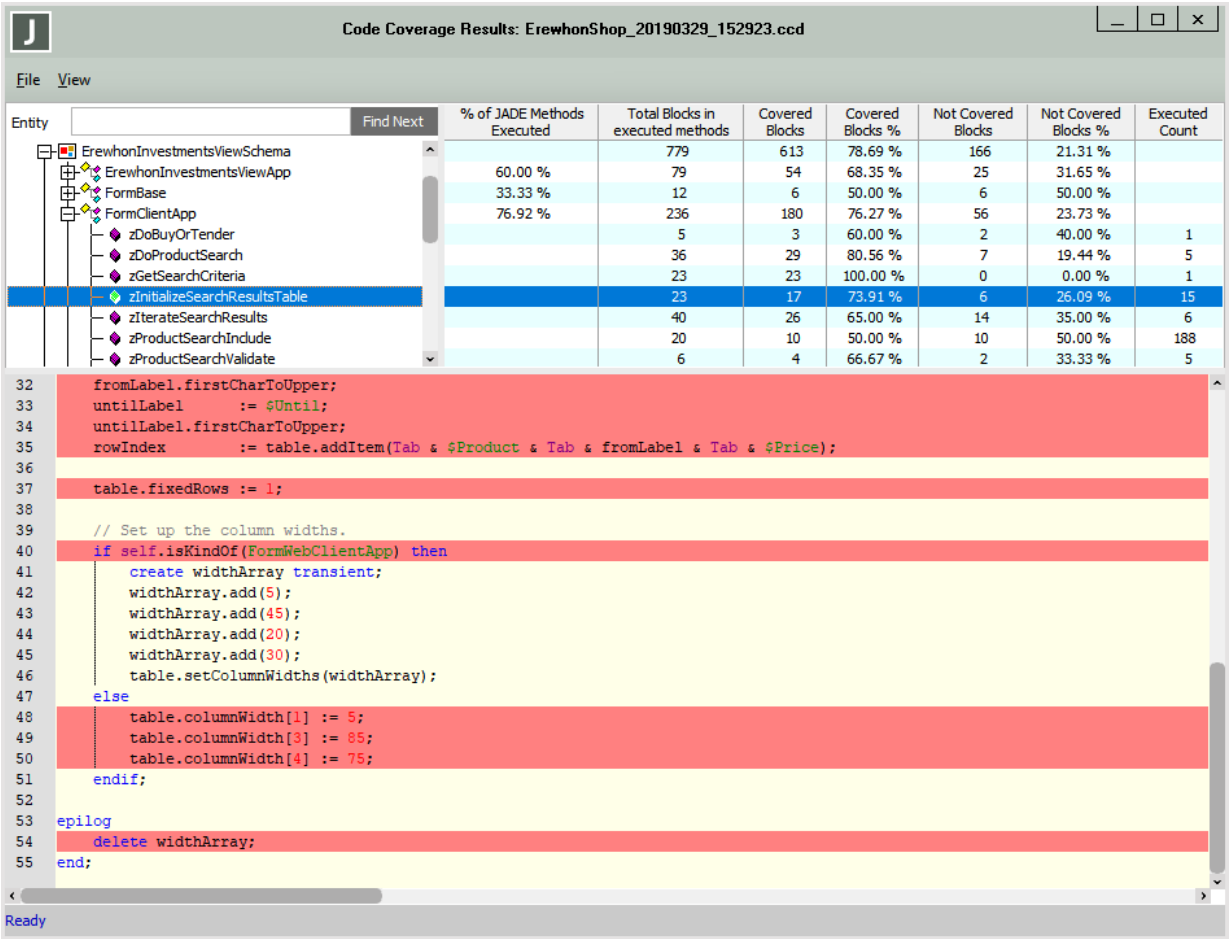
- **View** command in the Code Coverage menu of the Jade system tray.

  The current code coverage session is then stopped, the covered report file generated, and the **CodeCoverage** application is initiated so that that file is automatically loaded.

For more details, see the following subsections.

Code Coverage Results Browser Information

The following image is an example of loading code coverage information for the **Erewhon** example schema.



The top-left pane displays the schema entities for which coverage information has been recorded (in the file or files that have been loaded). Hierarchical and *flat* (that is, schema, class, method) views are available from the View menu. If you enter text into the text box above the list, the first entry in the list that contains that string is selected (the search is not case-sensitive). You can use the **Find Next** button to find the next entry in the list that contains that text. If the string cannot be found, a message to this effect is displayed in the status bar at the bottom of the browser.

The top-right pane displays coverage information for the entity selected in the top-left pane, and details about that entity in the lower pane.

In the hierarchical view, each parent level displays consolidated coverage information for all of its child levels (for example, the top coverage element displays overall coverage information, each schema element displays coverage information for all of its child classes, and each class element displays coverage information for all of its child methods). In the hierarchy view, entries (levels) are added and removed in the list box dynamically as you expand and collapse levels in the tree. This enables the list box to handle files with a large number of items; potentially more items than can be displayed in a list box.

In the flat views, each element displays coverage information for itself only (that is, there is no roll-up of information). The flat method view can display a maximum of 32,000 items. If any view exceeds the maximum number of entries, a message box is displayed and the view displays up to the maximum number of entries only.

The columns in the top-right pane are as follows.

- Percentage of Jade methods executed for a class or schema

  This is the percentage of the number of Jade methods defined on the class or schema that were executed. The entry is empty if the list item is not a class or schema. You must manually request the percentage value for a schema (for details, see the File menu commands under "Menu Items", later in this document).

- Total blocks in executed methods

  This is the total number of blocks that are available for execution by the executed methods for the selected element. (Each method consists of a number of executable blocks.) Note that for schema, class, and primitive type entries, this does not include the number of blocks for methods that have not yet executed.

- Covered blocks

  This is the number of blocks that have been executed at least once. (If a single block is executed multiple times, it is still counted as one covered block only.)

- Covered blocks %

  This is the number of covered blocks as a percentage of the total blocks. Note that for schema, class, and primitive type entries, this does not include the number of blocks for methods that have not yet executed.

- Not covered blocks

  This is the number of blocks that have not executed at all in the executed methods (that is, the number of total blocks less the number of covered blocks).

- Not covered blocks %

  This is the number of blocks not executed in executed methods as a percentage of the total blocks.

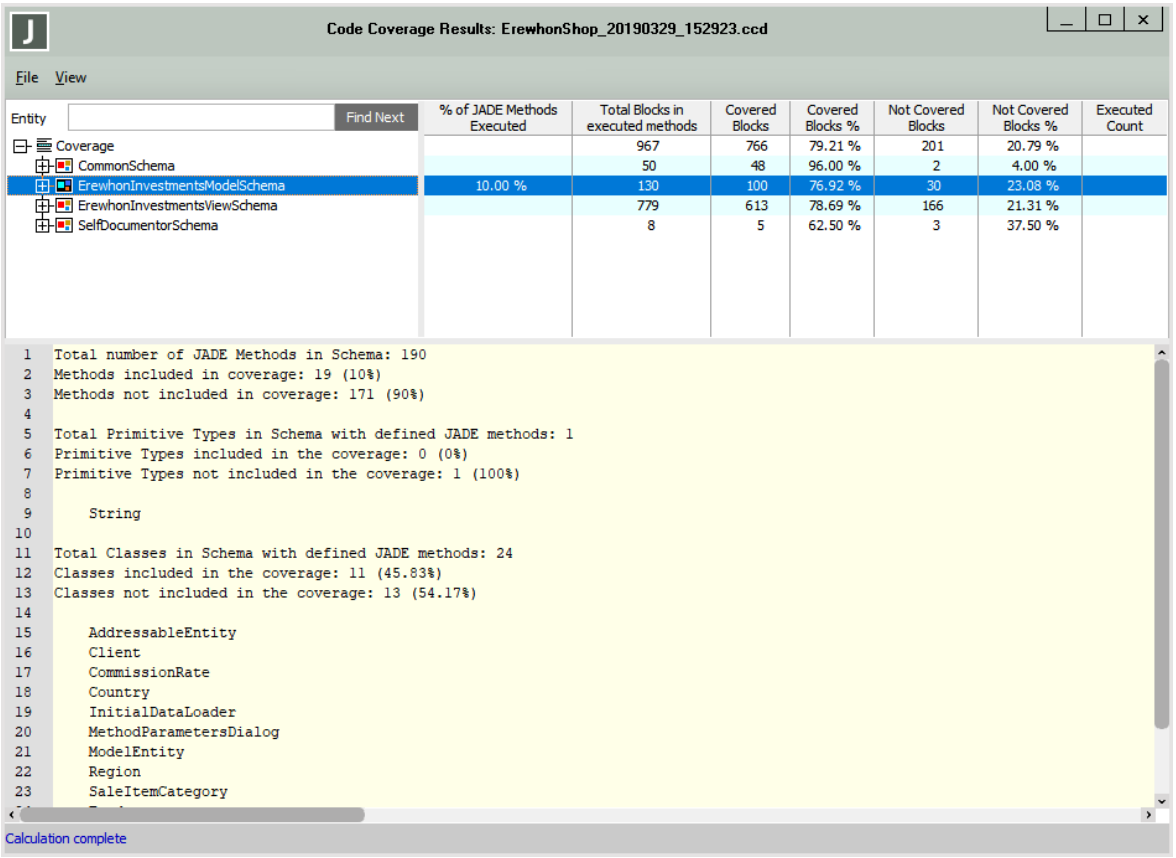
- Executed count

  This is the total number of times a method was executed.

Bottom Pane Display

The bottom pane displays the following information when the element selected in the top-left pane is:

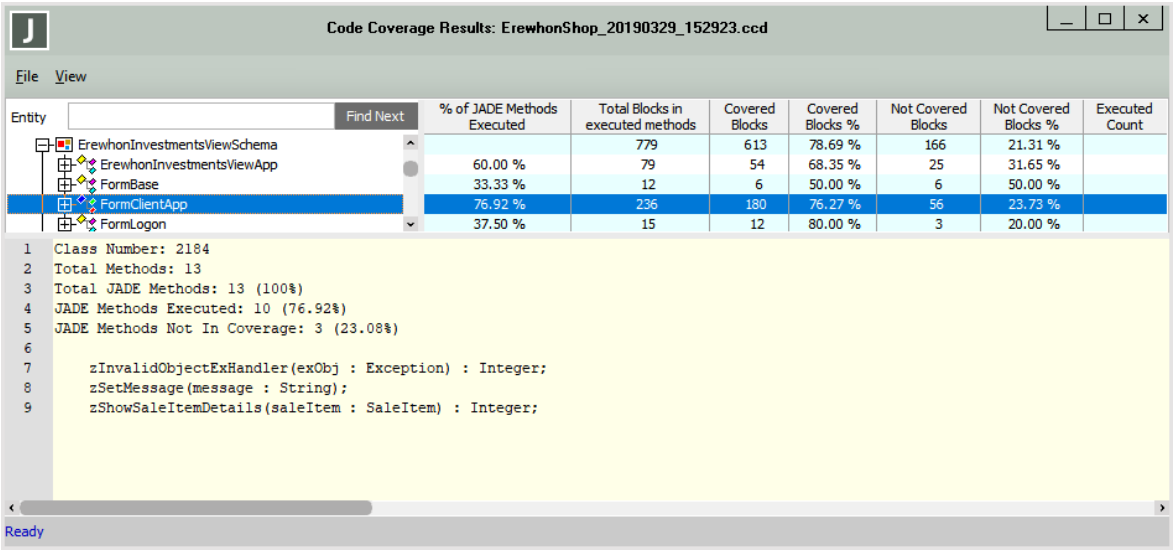
- The top-level coverage item, the bottom pane displays nothing.
- A schema, the bottom pane displays overall class coverage information for that schema, as shown in the following image.



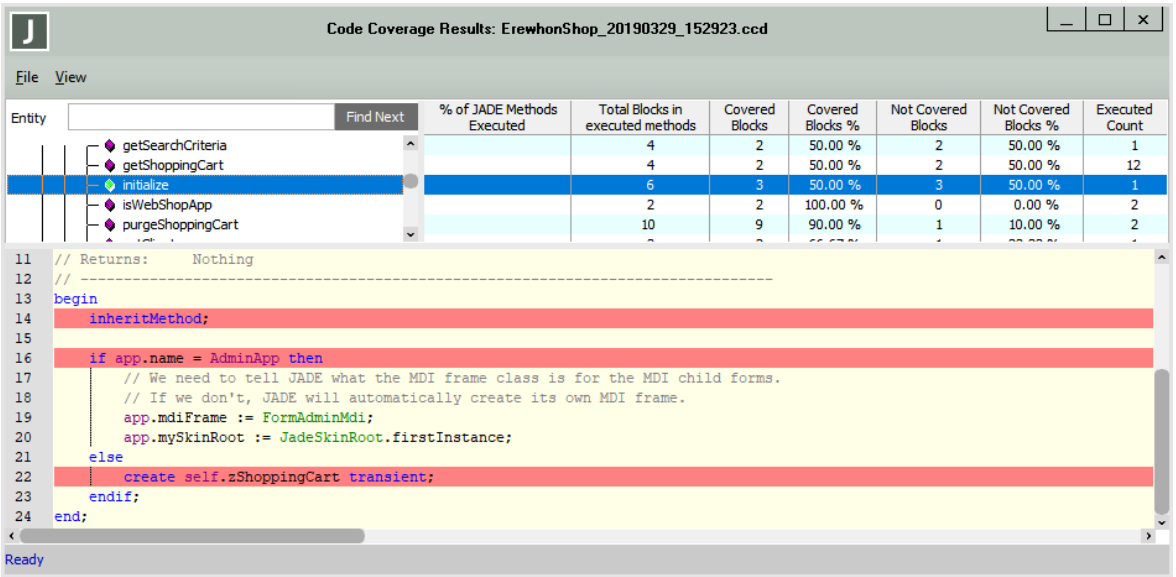
Note The figures for the total number of Jade methods in the schema, the methods included in the coverage, and those not included in the coverage are displayed only after you have selected the **Calculate Total Methods Executed Percent for Schema** command from the File menu for that schema entity.

- When the selected element is a class, the bottom pane displays overall coverage information for that class, including the Jade methods (that is, excluding any external methods) that do not appear in the code coverage results.

For example, in the following image, three Jade methods on the **FormClientApp** class are not displayed in the code coverage results (that is, they have not been executed).



- When the selected element is a method, the bottom pane displays the method source with a pale red background color indicating the lines of code that have been covered. As the following image shows, a line is highlighted if it contains at least one block that has been executed. Currently there is no indication of partially executed lines (that is, where a line contains multiple blocks but only some of the blocks have been executed). A line of code is highlighted entirely or not at all.

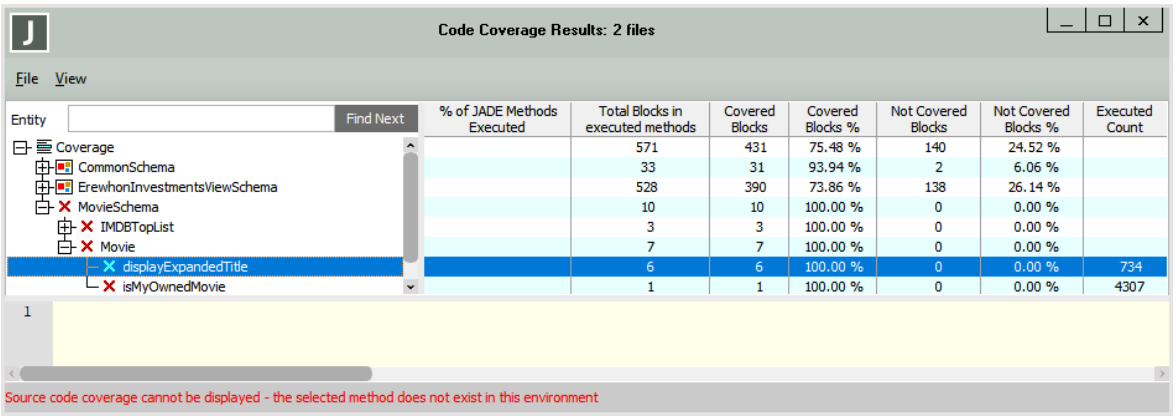


The Code Coverage Results Browser can load coverage output files that have been captured in any environment, regardless of whether or not the schema entities (that is, schemas, classes, and methods) exist in the environment from which the Code Coverage Results Browser is running. This means that potentially the browser may be displaying information for entities that it cannot find or entities that are different from the entities for which the output was recorded.

When a file is loaded, the following checks are performed.

- The browser attempts to find an entity using its fully qualified name (that is, it attempts to find methods using the *schema-name*::*class-name*::*method-name*).
- If an entity cannot be found by name, it does not exist in the browser's environment and is displayed in the top-left pane with a cross icon indicating this.

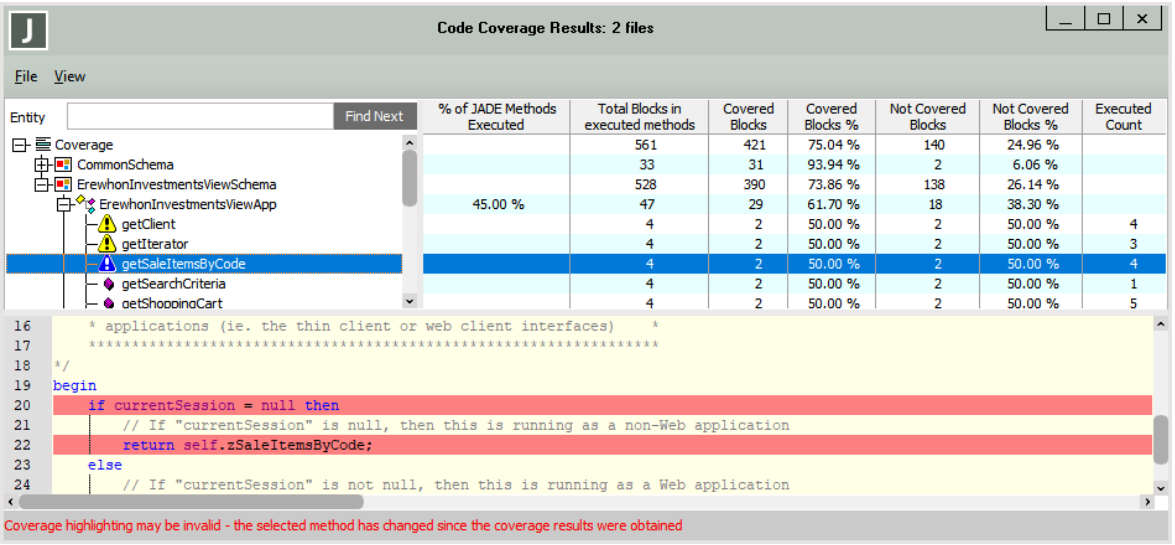
  Coverage information for such items is still displayed in the top-right pane because this information comes entirely from the coverage output file, but no additional information for such items can be displayed in the bottom pane (an appropriate status line message indicates this). For example, in the following image, the **Movie** schema does not exist in the browser's environment, so all of its child items are marked with a cross and the bottom pane does not show any source code coverage for the selected **displayExpandedTitle** method.



- If an entity is found using the fully qualified name, the object identifier (oid) and edition of the entity are compared against the object identifier and edition recorded in the coverage output file (that is, the entity in the browser's environment is checked against the object identifier and edition of the entity for which the coverage information was recorded).

  If they both match, everything is in order so all coverage information for this entity is displayed (that is, the bottom pane displays overall class coverage information for a selected schema element, as is shown in the first image in this section).
- If the object identifier or the edition do not match, the entity is different from the one for which coverage results were recorded; for example, a method may have been changed and recompiled since the coverage information was captured or coverage information obtained in an environment running a later version of a schema may be loaded into a Code Coverage Results Browser running from an environment containing an older version of the schema.

In these cases, entities are displayed in the top-left pane with an exclamation mark icon indicating this. All coverage information is still displayed, including the source code information in the bottom pane. However, the source code highlighting may be invalid if the method has been changed significantly since the coverage information was recorded (an appropriate status line message is given in these cases). For example, the following image shows that the **getSalesItemByCode** method exists in the browser's environment but it is different from the method in the coverage file.



Menu Items

This section describes the Code Coverage Results Browser menus and their commands.

- Load Submenu Load and Replace Coverage for Methods
- Load Submenu Load and Merge Coverage for Methods
- Clear Session
- Save As CSV
- Save Schema Methods Not in Coverage as CSV
- Calculated Total Methods Executed Percent for Schema
- Exit
- Hierarchy
- Schemas Only
- Classes Only
- Methods Only
- Sort By Name
- Sort By Total Blocks

- Sort By Covered Blocks
- Sort By Covered Blocks Percentage
- Ascending
- Descending
- Display List of Entities Not Covered

Load Submenu Load and Replace Coverage for Methods

When you select the **Load** command in the File menu and then select the **Load and Replace Coverage for Methods** command from the Load submenu, a dialog is displayed that enables you to select one or more code coverage output files to be loaded. The information in the file or files will be merged with the currently loaded information (assuming there is some). Progress information is displayed in the status bar and a button on the dialog enables you to cancel the load.

When merging a file (that is, loading a file into a non-empty session), coverage information may be being loaded for methods for which coverage information exists in the current session. In such cases, method editions are used to determine which has the latest coverage information: the current session or the file being loaded.

If the method edition in the currently loaded information is greater than the method edition in the file, the current session is assumed to already have the latest coverage information so the data in the file (for this method only) is ignored.

If the method edition in the file is greater than or equal to the edition of the method in the current session, the file is assumed to contain later information so the current session information (for this method only) is discarded and recreated from the file being loaded.

Load Submenu Load and Merge Coverage for Methods

The **Load and Merge Coverage for Methods** command from the Load submenu behaves the same as the **Load and Replace Coverage for Methods** command documented in the previous section, except that if coverage information for a method and the method editions are the same, the new coverage information for the method is merged with the existing information.

Clear Session

The File menu **Clear Session** command clears the current session information so that a new code coverage output file can be loaded from scratch.

Save As CSV

The File menu **Save As CSV** command displays the common Save As dialog, which prompts you to specify a file name and then saves the current session information to this file in CSV format. Regardless of the selected view, the information is written to the CSV file in hierarchy order, with the following columns: **Schema**, **Type**, **Method**, **Total Blocks**, **Covered Blocks**, **Covered Blocks %**, **Not Covered Blocks**, and **Not Covered Blocks %**.

The **Type** column can contain a class or a primitive type entity.

Save Schema Methods Not in Coverage as CSV

The File menu **Save Schema Methods Not in Coverage as CSV** command is available only when a schema entity is currently selected.

The common Save As dialog is then displayed, prompting you for a file name and the file path, if you want it saved to a location other than the default **bin** directory. A list of all Jade methods in the selected schema that have not been executed is then saved to the CSV file with the following columns: **Schema**, **Type**, and **Method**.

The **Type** column can contain a class or a primitive type entity.

After the file has been produced, the total number of Jade methods in the schema, the number of methods included in the coverage, and the number of methods not included in the coverage is included in the display in the pane at the bottom left of the Code Coverage Results Browser. In addition, the total percentage of Jade methods executed is then listed in the table. These values are cleared if another file load is performed.

Calculated Total Methods Executed Percent for Schema

The File menu **Calculated Total Methods Executed Percent for Schema** command is available only when a schema entity is currently selected.

When you select this command, the total percentage of Jade methods executed for this schema is calculated and displayed in the table.

In addition, the total number of Jade methods in the schema, the number of methods included in the coverage, and the number of methods not included in the coverage is included in the display in the pane at the bottom left of the browser.

These values are cleared if another file load is performed.

This step is manual because this calculation could take a significant amount of time for a large schema.

Exit

The File menu **Exit** command terminates the **CodeCoverage** application.

Hierarchy

The View menu **Hierarchy** command selects the hierarchy view, which is the default view.

Schemas Only

The View menu **Schemas Only** command selects a flat view, showing schema items only.

Classes Only

The View menu **Classes Only** command selects a flat view, showing class items only.

Methods Only

The View menu **Methods Only** command selects a flat view, showing method items only.

Sort By Name

The View menu **Sort By Name** command is valid only when one of the flat view commands has been selected.

This command sorts the list by name, in ascending or descending order (depending on whether the **Ascending** or **Descending** command is selected in the View menu).

Sort By Total Blocks

The View menu **Sort BY Total Blocks** command is valid only when one of the flat view commands has been selected.

This command sorts the list by total blocks, in ascending or descending order (depending on whether the **Ascending** or **Descending** command is selected in the View menu).

Sort By Covered Blocks

The View menu **Sort By Covered Blocks** command is valid only when one of the flat view commands has been selected.

This command sorts the list by covered blocks, in ascending or descending order (depending on whether the **Ascending** or **Descending** command is selected in the View menu).

Sort By Covered Blocks %

The View menu **Sort By Covered Blocks %** command is valid only when one of the flat view commands has been selected.

This command sorts the list by the covered blocks percentage, in ascending or descending order (depending on whether the **Ascending** or **Descending** command is selected in the View menu).

Ascending

The View menu **Ascending** command is valid only when one of the flat view commands has been selected.

This command sorts the list in ascending order.

Descending

The View menu **Descending** command is valid only when one of the flat view command has been selected.

This command sorts the list in descending order.

Display List of Entities Not Covered

The View menu **Display List of Entities Not Covered** command toggles (that is, enables or disables) this command, which is enabled by default.

This command controls whether the list of entities not in the coverage is displayed when a schema, class, or primitive type is currently selected.

When you click on a:

- Schema entity and the command is enabled, the list of classes with Jade methods that have not been included in the coverage is displayed.
- Class or primitive type entity and the command is enabled, the list of Jade methods that have not executed is displayed.