Run/Debug Configuration: Grails

Grails run/debug configuration enables you to run and debug the Grails applications, tests and Web tests.

The dialog box consists of the following tabs:

- Grails tab
- Code Coverage tab
- Maven Settings tab

This section provides descriptions of the configuration-specific items, as well as the toolbar and options that are common for all run/debug configurations.

Grails tab

Item	Description
Module	Select application, for which this run/debug configuration is created. By default, the name of the current module is suggested.
Command line	Type a command to execute a particular target, for example, `run-app`, or `app-engine`. Alternatively, you can execute target as described in the section Running Grails Targets.
VM Options	Specify the string to be passed to the VM for launching the application. This string may contain the options such as `-mx`, `-verbose`, etc. When specifying the options, follow these rules: ▪ Use spaces to separate individual options, for example, `-client -ea -Xmx1024m`. ▪ If an option includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, `some" "arg` or `"some arg"`. ▪ If an option includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, `-Dmy.prop=\"quoted_value\"`.
Environment Variables	Click the **Browse** button ... to open the **Environment Variables** dialog box, where you can create variables and specify their values.
Add --classpath	If this check box is selected, it means that the user intends to include the dependency directly, by passing `--classpath` to the command line. Please note the following: ▪ This check box is enabled, if a module has a dependency that was added directly from the Dependencies tab of the Project Structure dialog, without changing `BuildConfig.groovy` or plugins dependencies. ▪ Read about adding module dependencies in the section Configuring Module Dependencies and Libraries.

Item	Description
Launch browser	Select this check box to open the default browser.

Code Coverage tab

Use this tab to configure code coverage monitoring options.

Item	Description
Choose code coverage runner	Select the desired code coverage runner. By default, IntelliJ IDEA uses its own coverage engine with the Sampling mode. You can also choose JaCoCo or Emma for calculating coverage.
Sampling	Select this option to measure code coverage with minimal slow-down.
Tracing	Select this option to collect accurate branch coverage. This mode is available for the IntelliJ IDEA code coverage runner only.
Track per test coverage	Select this check box to detect lines covered by one test and all tests covering line. If this check box is selected, becomes available on the toolbar of the coverage statistic pop-up window. This option is only available for the *Tracing mode* of code coverage measurement for the testing run/debug configurations. Refer to the section Viewing Code Coverage Results.
Merge data with previous results	When you run your unit testing or application configuration several times, use this item to calculate statistics in the **Project View**, taking into account the statistics of each time you have run the configuration. Finally, the line is considered *covered* if it is covered at least once.
Packages and classes to record code coverage data	Click and buttons to specify classes and packages to be measured. You can also remove classes and packages from the list by selecting them in the list and clicking the — button.
Enable coverage in test folders.	If this check box is selected, the folders marked as test are included in the code coverage analysis.

Maven settings tab

Item	Description
Work offline	If this option is checked, Maven works in offline mode and uses only those resources that are available locally. This option corresponds to the `--offline` command line option.

Item	Description
Use plugin registry	Check this option to enable referring to the Maven Plugin Registry. This option corresponds to the `--no-plugin-registry` command line option.
Execute goals recursively	If this option is cleared, the build does not recur into the nested projects. Clearing this option equals to `--non-recursive` command line option.
Print exception stack traces	If this option is checked, exception stack traces are generated. This option corresponds to the `--errors` command line option.
Output level	Select the desired level of the output log, which allows plugins to create messages at levels of *debug*, *info*, *warn*, and *error*, or disable output log.
Checksum policy	Select the desired level of checksum matching while downloading artifacts. You can opt to fails downloading, when checksums do not match (`--strict-checksums`), or issue a warning (`--lax-checksums`).
Multiproject build fail policy	Specify how to treat a failure in a multiproject build. You can choose to: - Fail the build at the very first failure, which corresponds to the command line option `--fail-fast`. - Fail the build at the end, which corresponds to the command line option `--fail-at-end`. - Ignore failures, which corresponds to the command line option `--fail-never`.
Snapshot update policy	Specify whether the snapshot dependencies should be updated.
Plugin update policy	Select plugin update policy from the drop-down list. You can opt to: - Check for updates, which corresponds to the command line option `--check-plugin-updates`. - Suppress checking for updates, which corresponds to the command line option `--no-plugin-updates`.
Maven home directory	By default, this field shows the fully qualified name of the Maven installation directory. If you need to specify another directory, check the **Override** option, click the ellipsis button and select the desired path in the **Select Maven Installation Directory** dialog.
User settings file	Specify the file that contains user-specific configuration for Maven in the text field. If you need to specify another file, check the **Override** option, click the ellipsis button and select the desired file in the **Select Maven Settings File** dialog.
Local repository	By default, the field shows the path to the local directory under the user home that stores the downloads and contains the temporary build artifacts that you have not yet released. If you need to specify another directory, check the **Override** option, click the ellipsis button and select the desired path in the **Select Maven Local Repository** dialog.

Toolbar

Item	Shortcut	Description
+	`Alt+Insert`	Click this button to add new configuration to the list.
−	`Alt+Delete`	Click this button to remove the selected configuration from the list.
⎘	`Ctrl+D`	Click this button to create a copy of the selected configuration.
🔧	Edit defaults	Click this button to edit the default configuration templates. The defaults are used for the newly created configurations.
⬆ or ⬇	`Alt+Up` or `Alt+Down`	Use these buttons to move the selected configuration or group of configurations (folder) up and down in the list. The order of configurations or folders in the list defines the order, in which configurations appear in the **Run/Debug** drop-down list on the main toolbar.
📁		Use this button to create a new folder. If one or more run/debug configurations have the focus, then the selected run/debug configurations are automatically moved to the newly created folder. If only a category has the focus, then an empty folder is created. Move run/debug configurations to a folder using drag-and-drop, or ⬆ ⬇ buttons.

Common options

Item	Description
Name	In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations.
Defaults	This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations.
Share	Select this check box to make the run/debug configuration available to other team members. If the directory-based project format is used, the settings for a run/debug configuration are stored in a separate xml file in the `.idea\runConfigurations` folder if the run/debug configuration is shared and in the `.idea\workspace.xml` file otherwise. If the file-based format is used, the settings are stored in the `.ipr` file for shared configurations or in the `.iws` file for the ones that are not shared. This check box is not available when editing the run/debug configuration defaults.

Item	Description
Before launch	Specify which tasks should be carried out before starting the run/debug configuration. The specified tasks are performed in the order that they appear in the list.

Item	Keyboard shortcut	Description

Item	Description

Item	Keyboard shortcut	Description
+	`Alt+Insert`	Click this icon to add a task to the list. Select the task to be added: ■ **Run External tool.** Select this option to run an application which is external to IntelliJ IDEA. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in IntelliJ IDEA yet, add its definition. For more information, see Configuring Third-Party Tools and External Tools. ■ **Make.** Select this option to have the project or module compiled. The Make Module command will be carried out if a particular module is specified in the run/debug configuration, and the Make Project command otherwise. If an error occurs during the compilation, IntelliJ IDEA won't attempt to start the run/debug configuration. ■ **Make, no error check.** The same as the **Make** option but IntelliJ IDEA will try to start the run/debug configuration irrespective of the compilation result. ■ **Build Artifacts.** Select this option to have an artifact or artifacts built. In the dialog that opens, select the artifact or artifacts that should be built. See also, Configuring Artifacts. ■ **Run Ant target.** Select this option to have an Ant target run. In the dialog that opens, select the target to be run. For more information, see Ant. ■ **Generate CoffeeScript Source Maps.** Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see CoffeeScript Support. ■ **Run Maven Goal.** Select this option to have a Maven goal run. In the dialog that opens, select the goal to be run. For more information, see Maven. ■ **Run Remote External tool**: Add a remote SSH external tool. Refer to the section Remote SSH External Tools for details.
−	`Alt+Delete`	Click this icon to remove the selected task from the list.

See Also

Concepts:

- Groovy

Procedures:

- Grails Procedures

Web Resources:

- Developer Community