

JetBRAINS

**TC** TeamCity



# TeamCity 1.0 Reviewer's Guide

## Your Invitation to Review

Thank you for your interest in reviewing JetBrains TeamCity 1.0, an innovative new team environment aimed at boosting the productivity of software development teams by removing typical bottlenecks in the development process that often bog teams down.

This guide is not intended as an in-depth roadmap to TeamCity's functionality. Rather, we will try to give you an overview of the product and the benefits it delivers in a way that hopefully will help you to write an informed review in a minimum amount of time.

The guide is divided into three parts. Part 1 is a broad overview of the product which many authors may find sufficient to enable writing a basic review or article. Parts 2 and 3 provide guidance for exploring TeamCity's administration and end-user features for those authors who may wish to write a more in-depth review.

Wherever relevant, you will find links to other available resources that can provide additional information should you want it.

## Support and Information

TeamCity's online Getting Started Guide (link below), plus embedded tips in the web interface can help you get started and answer many questions.

However, if you need technical or other assistance to facilitate your review, we'll be more than happy to assist you. Simply contact the JetBrains office nearest you.

### North America:

Brian Noll

[brain.noll@jetbrains.com](mailto:brain.noll@jetbrains.com)

+1 (609) 714-7883

### Europe:

Ann Oreshnikova

[editors@jetbrains.com](mailto:editors@jetbrains.com)

+7 (812) 380 16 41

## Useful Links

### TeamCity:

- Official web site: <http://www.jetbrains.com/teamcity/>
- Getting Started Guide: [http://www.jetbrains.com/teamcity/documentation/getting\\_started.html](http://www.jetbrains.com/teamcity/documentation/getting_started.html)
- TeamCity Online Test Drive: <http://teamserver.jetbrains.com/login.html>

### JetBrains:

- Company history: <http://www.jetbrains.com/company/jbstory.html>
- Key executive profiles: <http://www.jetbrains.com/company/people/executives.html>
- Industry Awards: <http://www.jetbrains.com/company/press/awards.html>

## PART 1: TeamCity Overview

---

This section provides a “nutshell” description of TeamCity 1.0 and short descriptions of each of the main feature areas, plus information about obtaining and licensing TeamCity.

### Basic Concepts

JetBrains' flagship product, IntelliJ™ IDEA, has been widely praised by both industry pundits and developers alike for the productivity gains it helps individual software developers achieve through its “intelligent assistant” approach which automates and/or eliminates many peripheral, redundant, time-consuming tasks that are necessary to handle, but that distract from the central objective of creating code.

TeamCity's fundamental premise is much the same – identify necessary but redundant or time-consuming things in the team development process that detract from overall team productivity, and then provide intelligent, automated solutions to reduce or eliminate human error, time-eating analyses, miscommunication, and other problems that can bog down teams and contribute to missed targets and poor morale.

The major problem areas that TeamCity addresses are:

- Continuous integration and testing
- Software build automation and management
- Team communication relating to both the above

Along with these issues, TeamCity provides integrated and automated solutions for code inspection and code coverage analysis that can serve to reduce the chances of problems occurring in the areas mentioned above. In addition, there are features for integrating and extending TeamCity in different environments. Following sections will provide an overview of all these feature areas.

Speaking of environments, TeamCity is independent of any IDE, and the server component can run on either Windows or UNIX/Linux. A web browser is the only client software required, although there is a plugin for IntelliJ IDEA that enables the IDE to serve as a TeamCity client providing the most-needed features for developers, plus some extended features (more about these later). For Windows users, there is also a separately-installable component that enables display of TeamCity notifications from the Windows system tray.

A complete rundown on system requirements is available in the online Getting Started Guide

([http://www.jetbrains.com/teamcity/documentation/getting\\_started.html](http://www.jetbrains.com/teamcity/documentation/getting_started.html)).

## How TeamCity Fits Together

The basic unit of work of TeamCity is a software *build*. To build your software using TeamCity, you need to have at least one computer (referred to as a “Build Agent”) with a set of predefined parameters (system and environment variables, JDK version, etc).

Each software build has its own *Build Configuration* – a set of actions and parameters that serve to control all aspects of the build. The Build Configuration specifies such things as build runner (Ant for example) and the build schedule (daily, weekly, specific time).

To be able to run a build, a Build Agent must meet the Build Configuration's requirements. If there is a *free* Build Agent matching the Build Configuration requirements at the time the build is triggered, the build is started. Otherwise, the build enters the *build queue* and is processed when a compatible build agent becomes free.

TeamCity server interacts with:

- The VCS (Version Control System) used to manage your project's source file revisions
- The Build Agents on which the builds run,
- A web browser in which you can monitor builds (among other things)
- An IDE where you modify your source code<sup>1</sup>

Like other JetBrains products, TeamCity is created by and for professional developers. Very early in the development cycle, JetBrains' own teams began using TeamCity every day to run IntelliJ IDEA, ReSharper, and even its own builds.

## Main Feature Areas

Let's proceed now to a quick look at each of the main feature areas of TeamCity 1.0.

### Continuous Integration and Testing

Most development organizations acknowledge the value of having continuous and automated test/build operations many times each day. Achieving this in real life, efficiently and without hassles, has proved another matter – until now.

With TeamCity, team members can *remotely* test and integrate their code changes many times per day. Once configured (configuration is intuitive and straightforward), TeamCity takes care of complex test running, error handling and notifications, and integration chores. All a developer needs to do is submit changes to version control and continue working. Tests run remotely so the developer's computer isn't bogged down. If tests fail, the developer is notified immediately and can take corrective action.

<sup>1</sup> A dedicated plugin is needed. Currently only IntelliJ IDEA 6.0 is supported, but plugins for other popular IDEs are planned in the near future.

TeamCity integrates with CVS, Subversion, and Perforce, automatically handling check-in of submitted code and updating sources for building.

## Software Build Automation and Build Management

This is another area that often mires team progress. First there's the issue of computing resources for building. TeamCity can leverage available computing resources of an entire organization with its innovative *Build Grid* concept which employs currently-unused resources from multiple computers, any of which can run multiple builds and/or tests at a time, for single or multiple projects.

Flexibility in building is the name of the game in TeamCity. You can define and run different build types (nightly, periodic, etc.) with different build configurations for any project, including customizable build triggers that provide unparalleled convenience and flexibility in managing your build process. You can set up builds to run on any number of computers (called *build agents*) anywhere on your network. When a build is triggered (either automatically or manually – which users can do any time) TeamCity processes it on any available agent for which there is a build configuration. If no resources are free, the build enters a queue and is processed automatically as soon as resources are available. All this can be easily monitored in the web interface.

Then there's the problem of failed builds and their aftermath. It's necessary to figure out what code broke the build, and who submitted problem code and when – usually a time consuming activity in all but the smallest teams. Once identified, the responsible developers have to be contacted, and they must fix bad code before the next build can happen. TeamCity all but eliminates this kind of morass.

When any submitted code breaks a build, the developer(s) who submitted it are immediately and automatically notified that their code may have broken the build. Team leaders and developers alike monitor build status in the web interface. For any failed build, they can check (in seconds, not hours) what changes were made by whom, both directly in the failed build, and since the last successful build.

## Team Communication

You know the scenario: the build is broken, but it takes a while for people to realize it, and then nobody is quite sure whose code broke it, but everybody assumes someone else is on top of the situation and that a fix will happen. Valuable time is lost sorting it all out. *No more of that with TeamCity.*

First of all, the customizable array of notifications (with delivery options that include e-mail, Jabber, the Windows task tray, and IDE messages) ensures that anyone who submitted code that could have broken the build is notified the moment something fails. The failure shows up immediately in the web interface so anyone on the team can see the status (and keep an eye on any or all builds).

Then there's a very handy feature called *Take responsibility*. If nobody has taken on responsibility for a fix, this fact is evident for all to see at a glance in the Web interface (and a couple of other possible ways). The pain of hapless developer

whose code broke the build is assuaged with a handy link that enables him/her to not only take on the fix, but to effortlessly inform the entire team that someone is working on it, and also let them know when the fix is complete.

## Server-side Code Analysis

In addition to having TeamCity run unit tests when creating builds, builds configured as an IntelliJ IDEA project file can also have TeamCity remotely run a set of IntelliJ IDEA code inspections. Any problems found are displayed in both web interface and the IntelliJ IDEA editor just like local inspections. (This capability is provided with a special integration plugin discussed a bit later). The inspections include checks for unused and unreachable code, declaration redundancies, and general inspection of implementation code.

## Code Coverage Analysis & Reporting

TeamCity can optionally run a code coverage analysis for Ant and/or IntelliJ IDEA projects to determine the extent to which project source code is covered by unit tests. Just check a checkbox when setting up a build configuration, and the build results page in the web interface will include an overview of coverage statistics and a link to a detailed report. This information can also be accessed in IntelliJ IDEA when using the available TeamCity plugin.

## Extension and Add-ons

TeamCity is extensible via a Java API (documentation is provided in the download archive). TeamCity is based on the open protocols, supports industry-standard VCSs, bug tracking tools, and, thus, guarantees no vendor lock-in. Anyone can easily develop and deploy custom server-side plugins and build agent plugins for the TeamCity:



JetBrains plans to release more TeamCity add-ons and extensions including integrations for popular IDEs such as Eclipse, NetBeans, and Visual Studio.

A couple of add-on extensions for TeamCity 1.0 are already available:

- Windows Tray notification client
- Integration plugin for IntelliJ IDEA.

Both add-ons can be downloaded from the *My Settings* page of the TeamCity web interface.

## Windows Tray Notifier

This separately-installed extension enables Windows users to receive TeamCity notifications as pop-up messages from the Windows system tray which can be reviewed any time via the tray icon (which also provides access to build status information).

You can define notification rules in your personal settings that specify what kind(s) of notifications you want to receive and how they should be delivered.

## TeamCity Plugin for IntelliJ IDEA

This free plugin is a must-have for users of IntelliJ IDEA (version 6.0 is required – versions pre-dating TeamCity can't support it). With it, an IDEA developer never needs to switch to a web browser. The most essential functionality for developers – initiating builds into the queue, viewing current builds and build/test results, for example – is accessible inside the IDE, including notifications and build results. The plugin adds the *TeamCity* item to the main menu and TeamCity icons to the status bar.

Using IDEA as a TeamCity client is of course efficient, but the plugin also provides some highly useful extended functionality that can help each developer minimize the possibility of submitting code that causes problems in the integration process.

### ***Personal Builds***

Developers using the TeamCity plugin can create a *personal build* which is run remotely on TeamCity, and is just the same as a “regular” build (including automatic integration of all the latest sources in version control) except that changed code is not committed to version control.

If a personal build fails, the “real” build is not affected because the code was not committed to version control, so there is no adverse impact on the team. The developer receives immediate notification and can then work on a fix, easily navigating to problem code right in the editor. Once the developer knows the code will build, he/she can commit it to version control and let the regular build process take its course, confident that there will not be any problems.

This feature, called *Remote Run*, significantly reduces the chances of any IntelliJ IDEA developer submitting build-breaking code.

### ***Pre-tested (Delayed) Commits***

It happens quite often: some member of the team submits changes that break the build, and goes home before it is discovered. This is a critical problem especially for geographically dispersed teams.

With TeamCity the software building process never “freezes”. Via IntelliJ IDEA the developers can perform “Pre-tested (Delayed) Commit” of the modified code. The developer's code modifications get integrated into the latest available sources of the project code in the VCS, and the tests are run remotely. If the tests pass, the modified code is *automatically* submitted to the VCS. And if the tests fail, the developer receives notifications, can fix the discovered problems in the IDE, but the *project code base stays clean*.

## Obtaining and Installing TeamCity

TeamCity is distributed as a Windows executable, a WAR archive, for installing into an existing Java EE container on your server, and as a Zip archive which

bundles Tomcat. All archives are available for download at

<http://www.jetbrains.com/teamcity/download/>.

Installation and startup are straightforward and completely explained in the detailed Getting Started Guide, available online at

[http://www.jetbrains.com/teamcity/documentation/getting\\_started.html](http://www.jetbrains.com/teamcity/documentation/getting_started.html).

## Licensing

TeamCity server requires no license to install and start up. Licenses are per user account on the server, and are needed in order for users to access TeamCity. A 60-day evaluation license providing full features and unlimited users is available and can be requested from the TeamCity download page.

User licenses are regularly priced at \$399 (US) per license. JetBrains is offering an introductory discount price of \$199 until December 1, 2006. Licenses may be purchased online at <http://www.jetbrains.com/teamcity/buy/>.

For a limited time JetBrains will include a free TeamCity 1.0 user license with every new or upgrade license for IntelliJ IDEA 6.0. This special deal enables entire teams to get the productivity benefits of TeamCity for just the cost of IntelliJ IDEA.

## Online Test Drive

JetBrains hosts a TeamCity installation on the web with sample projects that enable you to test-drive TeamCity and its web interface without any installation or setup. All you need is a user account which you can create immediately from the login screen. Access this hosted version at <http://teamserver.jetbrains.com/>.

## Need More?

This concludes Part 1 of the guide. Parts 2 and 3 provide pointers to help authors who wish to do an in-depth review to find and explore at various administrative and end-user features.

If you feel you have enough information at this point to be able write your review of TeamCity, feel free to stop here and start writing. And let us thank you again for your interest and remind you to get in touch if you have questions or need assistance.

If you want to explore TeamCity further, let's proceed to Part 2.

## PART 2:

# Administration and Configuration

---

For detailed information about working with TeamCity, please consult the online Getting Started Guide at

[http://www.jetbrains.com/teamcity/documentation/getting\\_started.html](http://www.jetbrains.com/teamcity/documentation/getting_started.html).

In this section we will point you to some of the main administration features that you need to use when getting started with TeamCity and from time to time afterwards.

## Accessing Administration

The first time you run TeamCity and access the login page, you will be prompted to create a new administration account. This process is brief and straightforward. The account you create will have full administration privileges for Team City.

When you log in with administration privileges, the *Administration* button appears in the web interface:



Clicking it will take you to the Administration page where the following administrative functions are available and/or accessible:

- View, create new, and edit existing projects.
- For each project, create new or edit existing Build Configurations
- Configure build history and history clean-up policy
- Create and manage User Accounts
- View and manage Licenses
- View and edit server configuration (which includes such items as authentication type, and hosts and ports for notification providers).

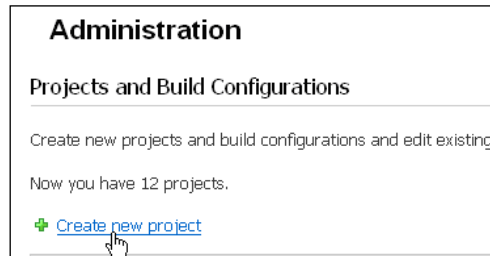
## Key Administration Operations

The administrative functions are generally easy to understand, but there are one or two that might need a bit of explanation.

### Setting up projects

A project consists of a set of VCS (version control system) roots which contain source code files, and one or more *Build Configurations*. A Build Configuration is a set of actions and parameters for creating application builds using the source code in the project roots.

To begin creating a project, use this link on the *Administration* page:



The web interface will walk you through a screen in which you specify the VCS roots for the project. Once the VCS roots are specified, you'll need to create at least one Build Configuration.

## Creating Build Configurations

A Build Configuration is essentially instructions for TeamCity that enable it to handle the chores involved in building the project's sources. Every project needs to have at least one Build Configuration defined. It's possible to create multiple configurations for different purposes. For example, in addition to your main build, you might create feature-limited builds, or builds to perform code coverage analysis, or builds for a specific customer.

To create a new Build Configuration, look for this link in the *Administration* page



or a selected project page:

Among the things you need to specify in a Build Configuration are:

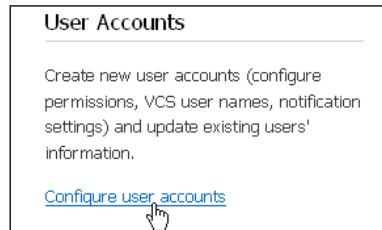
- Whether or not to automatically check out sources before building
- The build runner to use. TeamCity supports Java (Ant, Maven 2) and .NET build runners (NAnt, MS Build, and Visual Studio 2003 and 2005 solutions), and it can also build IntelliJ IDEA projects and create builds that also perform server-side code analysis<sup>2</sup>.
- Build file path
- JDK and JVM information
- Build numbering format and counter

## Setting Up User Accounts

You need a TeamCity license key for each user account you want to create. Some IntelliJ IDEA licenses allow the IntelliJ IDEA user access to TeamCity as well. When setting up users, make sure you have a the license key for all the user accounts you want to set up.

To begin working with user accounts, look for this link in the *Administration* page:

<sup>2</sup> The Ant build runner can also be used with Eclipse and NetBeans projects.



## Notification settings

If you want TeamCity to be able to deliver notifications by e-mail or Jabber, it's important to configure TeamCity to find the respective servers. You do this on the Server Configuration page. To get there, look for this link on the *Administration* page:



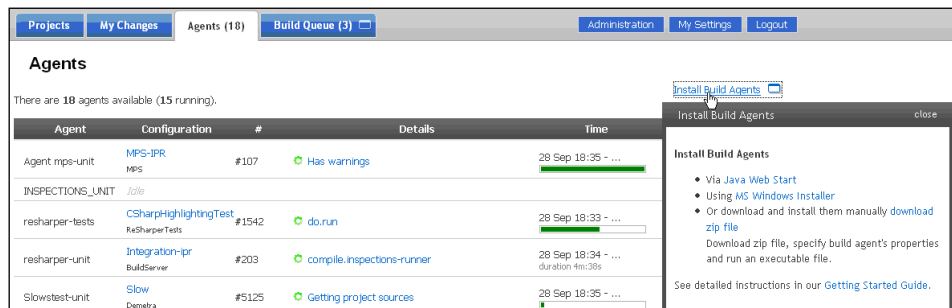
On the same page you can configure the authentication scheme. For more details see the Getting Started Guide.

## Installing Build Agents

This task does not require administration privileges, but it is something an administrator setting up a TeamCity installation might want to do. Alternatively, the task can be delegated to other TeamCity users.

Computers that can potentially run builds are referred to as *Build Agents*. They are part of the *Build Grid* – the pool of computing resources that can be used by TeamCity to run tests and builds. In this way, the workload of testing and building can be distributed among multiple computers and coordinated by TeamCity.

In order to allow a computer to act as a Build Agent, it must have Agent software installed. Using a web browser on the agent-to-be computer, any user can access the installer on the Agents tab of the web interface after logging in:



Agent	Configuration	#	Details	Time
Agent mps-unit	MPS-SPR MPS	#107	Has warnings	28 Sep 18:35 - ...
INSPECTIONS_UNIT	/Idle			
resharper-tests	CSharpHighlightingTest ReSharperTests	#1542	do.run	28 Sep 18:33 - ...
resharper-unit	Integration-pr BuildServer	#203	compile.inspections-runner	28 Sep 18:34 - ... duration 4m:38s
Slowtest-unit	Slow Demetra	#5125	Getting project sources	28 Sep 18:35 - ...

**Install Build Agents**

- Via Java Web Start
- Using MS Windows Installer
- Or download and install them manually [download zip file](#)

Download zip file, specify build agent's properties and run an executable file.

See detailed instructions in our [Getting Started Guide](#).

After you configure and run the Agent, it will register itself with the TeamCity server and can start running builds.

Note that the computer that hosts TeamCity server can also be an agent and run builds – just install the Build Agent as described above.

## PART 3: Typical User Tasks

In Part 3 of this guide, we'll briefly look at a number of typical ways you might interact with TeamCity and point you to the features for further exploration. We'll concentrate on the web interface since that is available to everyone regardless of what IDE they use.

### User Account Setup

The first thing a new user need to do is to set a few properties in his/her user account. After logging in, click the **My Settings** button:



On the My Settings page, you can specify such things as the name that appears to other TeamCity users, and you can also change your password. You can also control which project appear for you in the web interface – handy if there are a lot of projects and you work on only a few at a time. You can download the TeamCity plugin for IntelliJ IDEA and the Windows Tray Notifier extension on this page as well ([see Extension and Add-ons](#)).

The two most important configurations, however, are Version Control Settings and Notifications.

### Version Control Settings

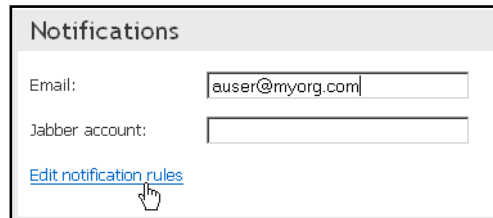
This is a very easy, but vital configuration. You need to specify your user name/ID for the version control systems you work with. This is so TeamCity can interact with the your VCS account, identifying your changes, and showing them in the *My Changes* page. You only need to specify a user name for the system(s) you actually use. The figure below shows the VCSs supported by TeamCity 1.0.

Version Control Settings	
CVS user:	<input type="text" value="auser"/>
Perforce user:	<input type="text"/>
Subversion user:	<input type="text" value="usera"/>

### Notification Settings and Notification Rules

Notification settings are very straightforward:

- If you want to receive TeamCity notifications by e-mail, provide your e-mail address.
- If you want to receive TeamCity notifications via Jabber, provide your Jabber account ID

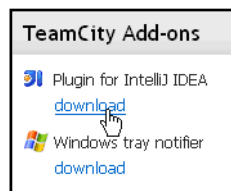


Notification rules enable you to specify what events you want to be notified about (a build starting to fail, for example) and how you want to be notified (by the Windows Tray Notifier, for example). You can specify different rules for builds containing your own changes, and for other builds.

## User Environment Setup

If you use the web interface to interact with TeamCity, there's really no setup needed. You just need a recent version of a mainstream browser. Supported browsers: MSIE 6.0, Mozilla Firefox 1.0 or later, Opera 8.5.x and 9.0, and Safari 2.0.x.

Windows users may want to install the Windows Tray Notifier, and IntelliJ IDEA users will almost certainly want to install the TeamCity plugin for IntelliJ IDEA. (We discussed these extensions in *Extension and Add-ons*.) Both can be downloaded from the *My Settings* page of the web interface:



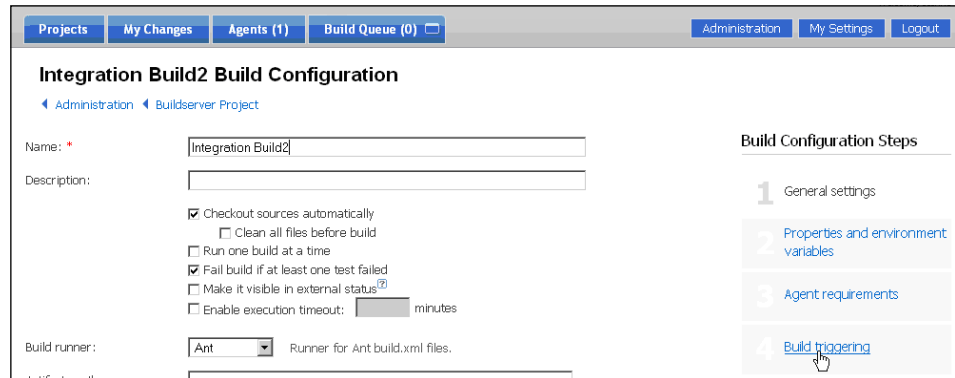
## Building Projects

This section will talk briefly about how to launch builds (automatically or explicitly), monitor builds, and view build results.

## Triggering Builds

A major advantage of TeamCity is the capability to trigger builds automatically at specific times or in response to specific events. However, there are times when you need to a launch build manually and this is fully supported and easy to do.

Before a build can be triggered a *Build Configuration* must be created as part of a project. (See Part 1, *Building and Build Management*). Properties that control automatic build triggering are specified in the Build Configuration's *Build Triggering* page, which is the fourth in the sequence of build configuration steps when creating a new configuration, or the fourth step accessible in the build configuration page:

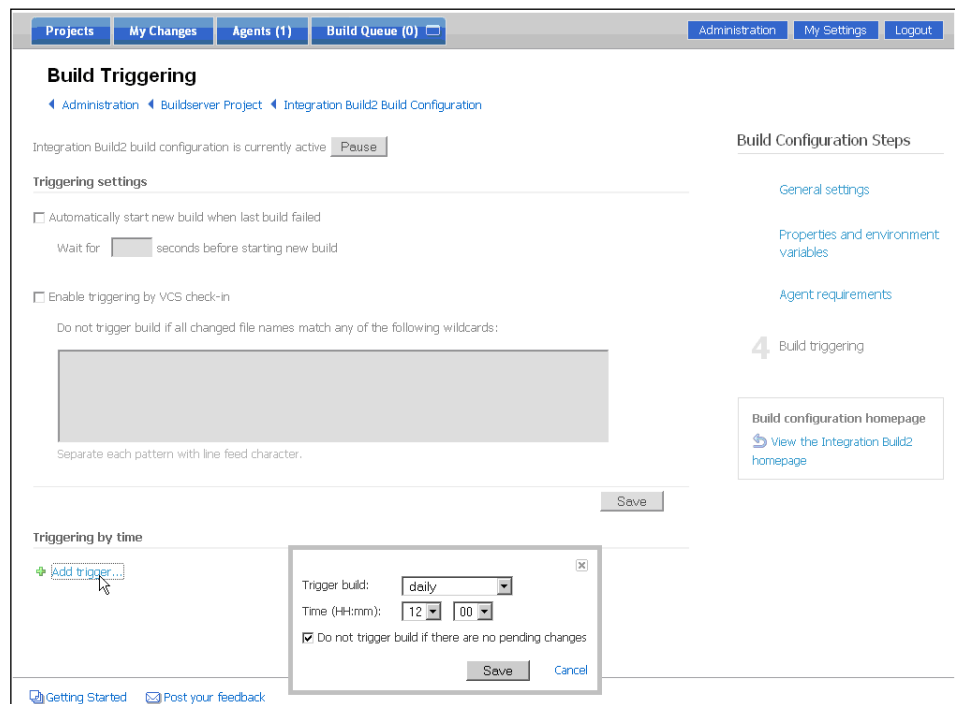


## Automatic Builds

The Build Triggering page of the Build Configuration (see following screenshot) enables you to control exactly when a build is triggered automatically.

### Automatic Triggering Options

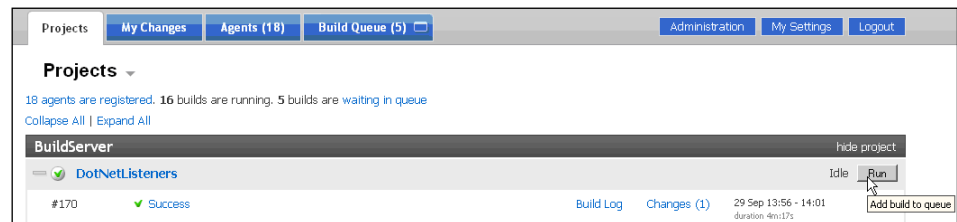
- You can opt to trigger a new build automatically after failure of the previous build, optionally specifying an interval of time to wait before starting the new build.
- Or you can opt to trigger a new build automatically whenever code is checked into the project's version control system, optionally specifying exceptions to this rule.
- Or you can opt to trigger a new build at some time interval (daily or weekly), specifying the time the build should begin, and optionally skipping building if there are no pending changes.



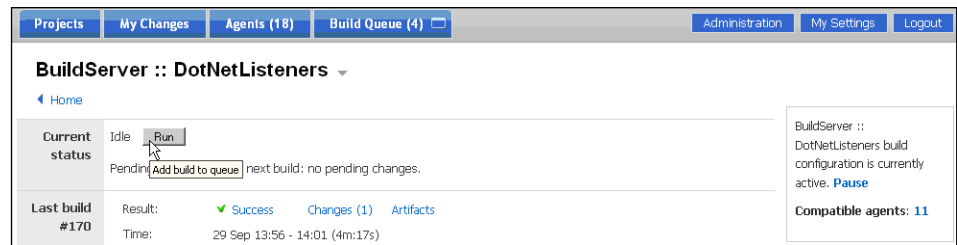
## Explicit Triggering

As mentioned, you can start a build any time. There are two main access points for this:

- The *Projects* tab. Click the **Run** button in the section of the project you want to build.



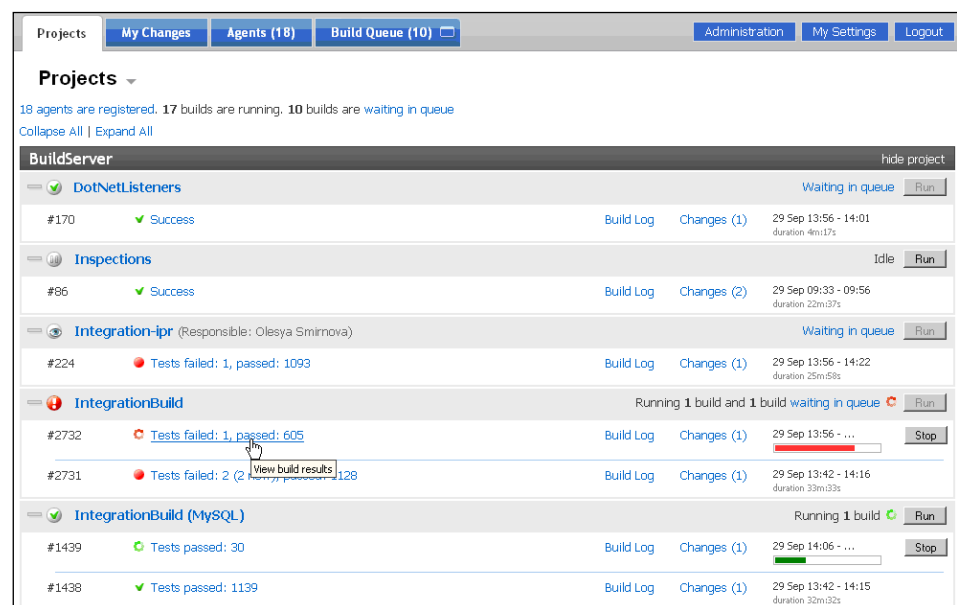
- The project's page (click it's name in the *Projects* page to go there). It also contains a **Run** button:



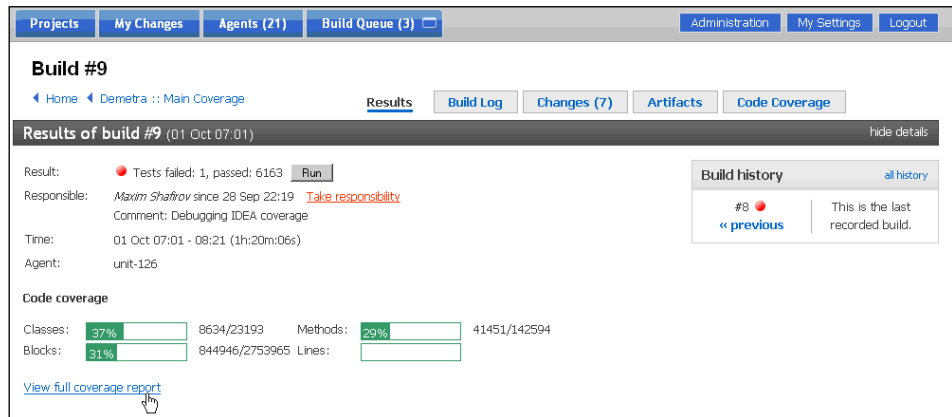
BUILDS can be explicitly triggered from the IDE when the TeamCity Plugin for IntelliJ IDEA is installed to the IDE.

## Monitoring Builds

It's quick and easy to check the status and results of any build for any project. Go to the **Projects** page and scroll if necessary to find the project that interests you. Recall that you can control the visibility of projects in *My Settings*.

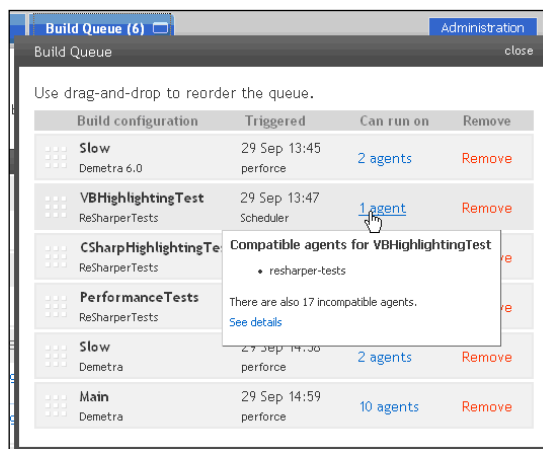


You can see detailed about the build by clicking the build status, which is a link to the Build Results page:



## Checking the Build Queue

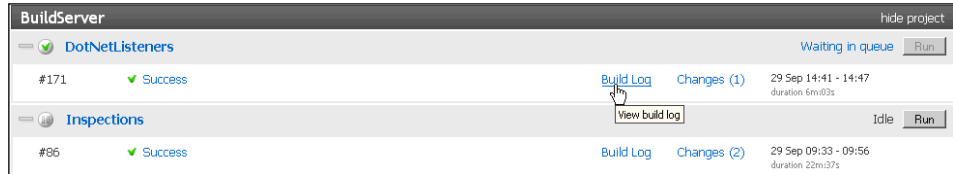
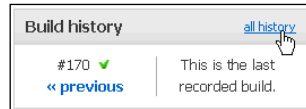
If no build agent is currently available to process a build when it is triggered, it enters the *Build Queue* where it awaits a free agent to process it. Click the **Build Queue** tab to see the queue.



You can optionally remove any build from the queue, see information about the potential agents that can run the build, or reorder the queue using drag-and-drop.

## Viewing the Build History

TeamCity maintains a complete history of builds which includes the artifacts produced by each build. History appears in the Build History section of the *Build Log* page of any build in the *Projects* tab.

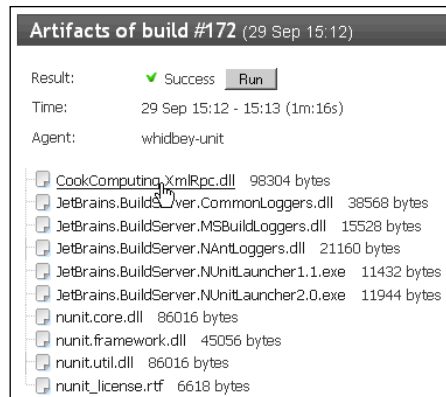



### All History

This link provides access to every build maintained in the history. Keep in mind that the history can be cleaned (in Administration) periodically. To prevent some build in the history from being cleaned (such as a milestone or released build), click the *Pin* link on the relevant build's row in the table of builds.

### Downloadable Artifacts

The Artifacts column of build history leads to a page where you can download the artifacts created by the build. You can also re-run the same build from that page:



The artifacts can be useful for any number of purposes in your organization: user testing, customer demos, documentation... it's up to you.

## Integrating Source Code Changes

As we've seen earlier, TeamCity can be set up to integrate and build changes often. When TeamCity creates a build, it automatically uses the latest source code in the project's version control system.

### Committing Code Changes

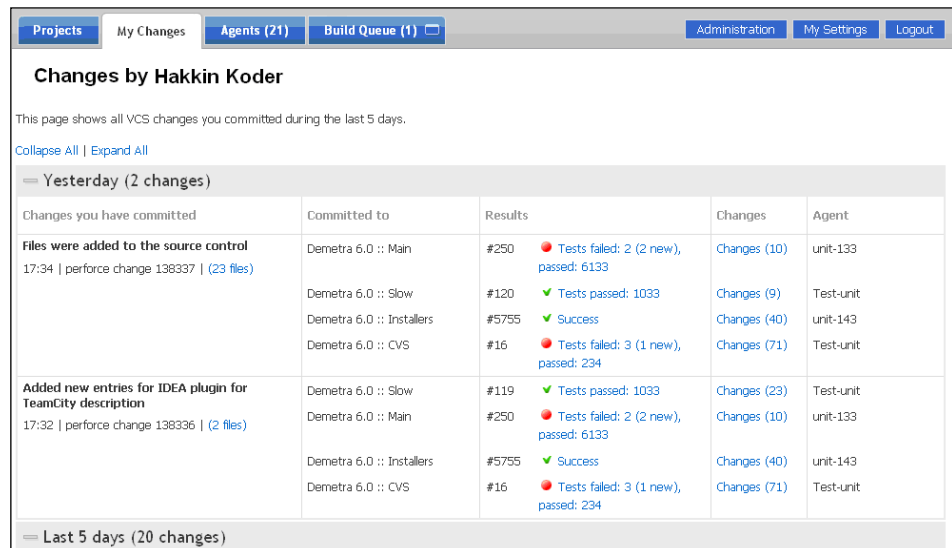
Those using TeamCity exclusively via the web interface simply submit their changes to version control as usual. TeamCity takes care of running unit tests and building the changes according to the applicable build configuration.

It's worth mentioning here that IntelliJ IDEA 6.0 users using the TeamCity plugin have additional options provided by the plugin, namely *Pre-tested Commit* and *Remote Run* which enable remote testing, integrating, and building of changed code without affecting the team's build. You can find more information about this feature in the online Getting Stared Guide (see "*Remote Run*" for testing your code).

## Viewing Your Own Changes

The *My Changes* tab in the web interface is provided to enable you to quickly check your recent changes in the project(s) you work on. There you can see:

- All changes you have checked in to version control recently
- Builds that contain changes you made, and the results of those builds



**Changes by Hakkin Koder**

This page shows all VCS changes you committed during the last 5 days.

[Collapse All](#) | [Expand All](#)

— Yesterday (2 changes)

Changes you have committed	Committed to	Results	Changes	Agent
<b>Files were added to the source control</b> 17:34   perforce change 138337   (23 files)	Demetra 6.0 :: Main	#250 <span style="color:red">●</span> Tests failed: 2 (2 new), passed: 6133	Changes (10)	unit-133
	Demetra 6.0 :: Slow	#120 <span style="color:green">●</span> Tests passed: 1033	Changes (9)	Test-unit
	Demetra 6.0 :: Installers	#5755 <span style="color:green">●</span> Success	Changes (40)	unit-143
	Demetra 6.0 :: CVS	#16 <span style="color:red">●</span> Tests failed: 3 (1 new), passed: 234	Changes (71)	Test-unit
<b>Added new entries for IDEA plugin for TeamCity description</b> 17:32   perforce change 138336   (2 files)	Demetra 6.0 :: Slow	#119 <span style="color:green">●</span> Tests passed: 1033	Changes (23)	Test-unit
	Demetra 6.0 :: Main	#250 <span style="color:red">●</span> Tests failed: 2 (2 new), passed: 6133	Changes (10)	unit-133
	Demetra 6.0 :: Installers	#5755 <span style="color:green">●</span> Success	Changes (40)	unit-143
	Demetra 6.0 :: CVS	#16 <span style="color:red">●</span> Tests failed: 3 (1 new), passed: 234	Changes (71)	Test-unit

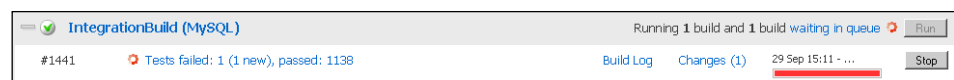
— Last 5 days (20 changes)

## Handling Problems

Fact of Life: failed tests and builds are going to happen in the process of developing anything non-trivial. The trick is to know as soon as possible when something fails, and then quickly pinpoint the code that caused a failure, understand who's code it is, get a fix going, and inform the team that the issue is being fixed. In this area, TeamCity really shines, saving a great deal of time and effort and keeping things on track.

## Failed Tests

The web interface immediately indicates in real time when any unit test fails:



IntegrationBuild (MySQL) Running 1 build and 1 build waiting in queue ● Run

#1441 ● Tests failed: 1 (1 new), passed: 1138 Build Log Changes (1) 29 Sep 15:11 - ... Stop

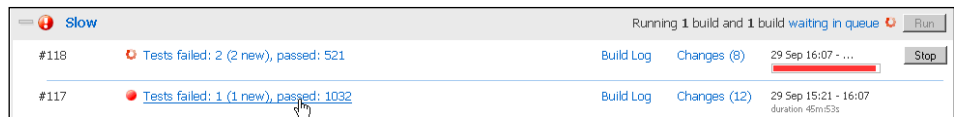
Furthermore, every developer who submitted code that failed a test are immediately notified by whatever means they have specified in their user settings (e-mail, Jabber, or whatever).

When tests start failing, any user other than a guest can stop the build, halting further tests from running, and/or the build from starting or processing further.

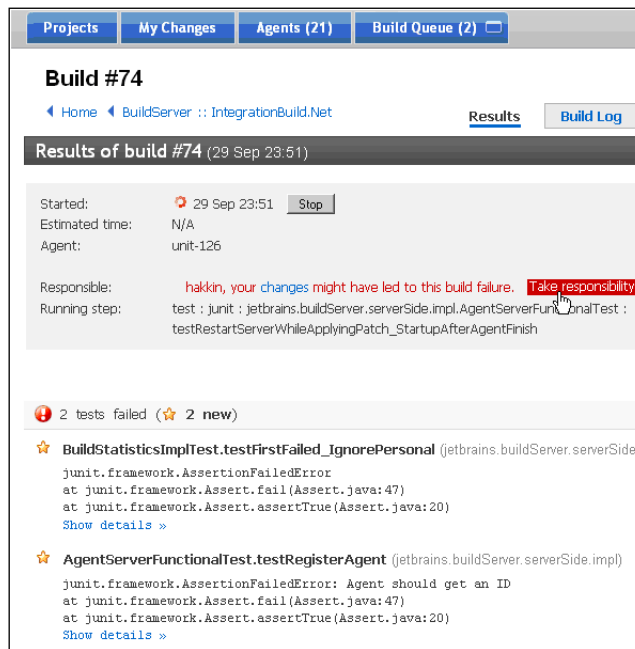
## Failed Builds

As with tests, your team knows immediately when a build fails... even when it *starts* failing. The web interface shows the status, and the people who have submitted code that's breaking things are notified, and a problem build can be stopped if it hasn't finished. So you should never have the scenario of a failed build and nobody knows about it.

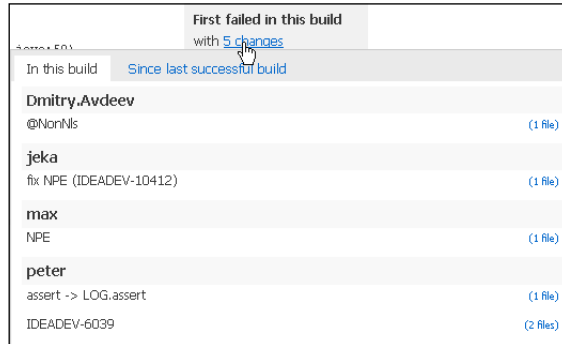
When the inevitable happens, it's quick and easy to zero in on the changes that probably caused the problem. Just click on this link in a failed build:



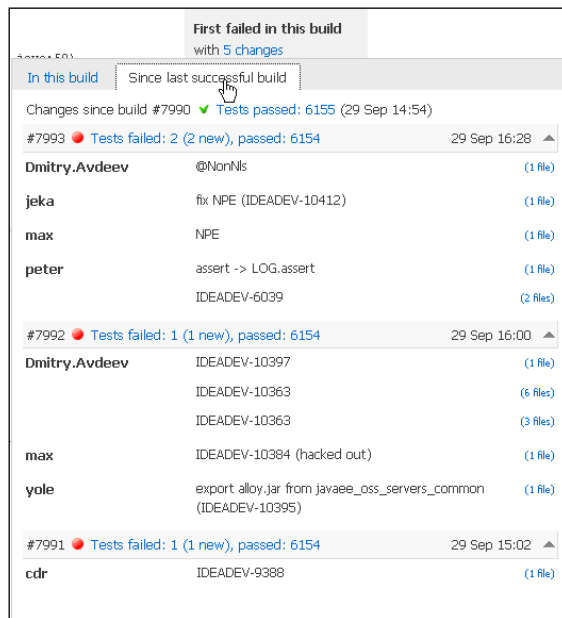
The build results page provides a wealth of information that can help team leads and members understand what happened.



The page provides detail about test failures, and makes it easy to explore the repository changes that are most likely involved. The page displays count of the number of source code changes in the build, and when you hover over that count, TeamCity pops up a list of those changes which includes the number of files changed, both in the current build...



...and since the last successful build:



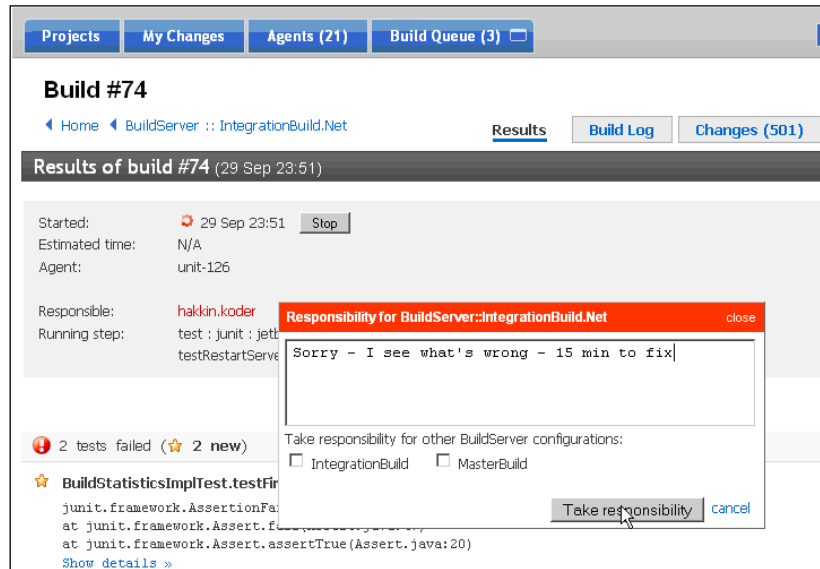
As you can see, these two views, accessed in seconds, significantly reduce the scope of possibilities for what code, and whose, broke the build.

### Taking Responsibility

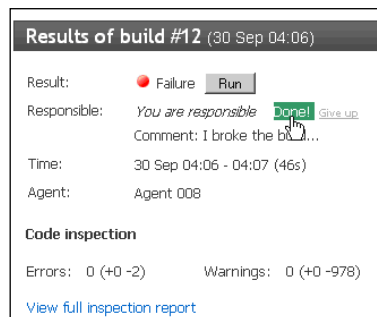
Another Fact of Life: sooner or later you will be the goat and something you submit will break a build. When it's your day in the barrel, TeamCity has a couple of handy features that make your life just a little easier in a time of stress.

First of all, you will most likely get the news from TeamCity first, and notification will happen as soon as the first breakage is detected. You may well have time to fix things before anyone else has to deal with the broken build.

Second, when it's clear that "you done it", there's a handy little link on the build results page:



This feature lets you claim responsibility for fixing the build, and notify the team (possibly including your boss!) that someone (namely you) has matters in hand, all in just a couple of seconds. And when you have fixed the problem, a similar link lets everyone know.



TeamCity takes care of the details via the web interface and notifications system. You spend your time on the fix.

## Using Code Quality Features

TeamCity has a couple of features that can help teams improve code quality and maintain higher code quality ongoing. Before we conclude this look at TeamCity, let's briefly look at these features:

- Server-side Code Analysis
- Code Coverage Analysis

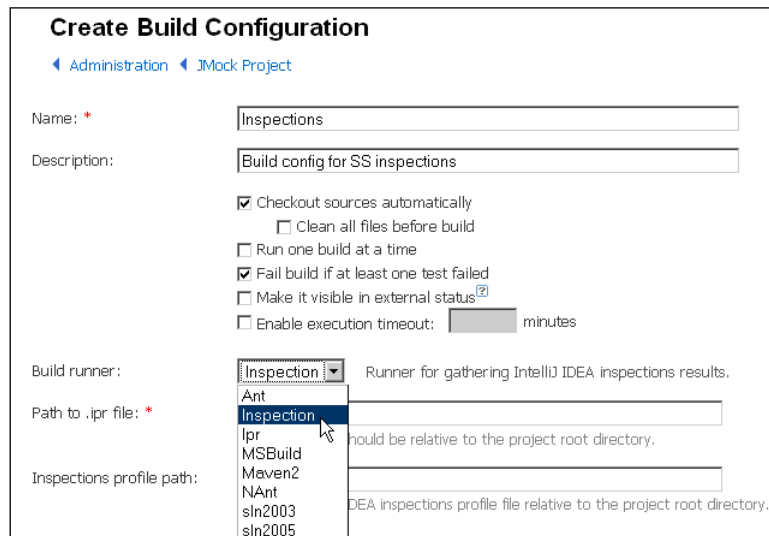
### Server-side Code Analysis

You can create a special Build Configuration that will:

- Perform a remote static code analysis and inspection of the code for unused and unreachable code
- Check for declaration redundancies
- Inspect the implementation code

Such build configuration will run a sub-set of IntelliJ IDEA's code inspections on the build sources. Any problems found are displayed in both web interface and the IntelliJ IDEA editor just like local inspections.

The key to setting up this build configuration is to select the **Inspection** option when specifying the build runner:



**Create Build Configuration**

Administration > JMock Project

Name: \*

Description:

Checkout sources automatically  
 Clean all files before build  
 Run one build at a time  
 Fail build if at least one test failed  
 Make it visible in external status<sup>(?)</sup>  
 Enable execution timeout:  minutes

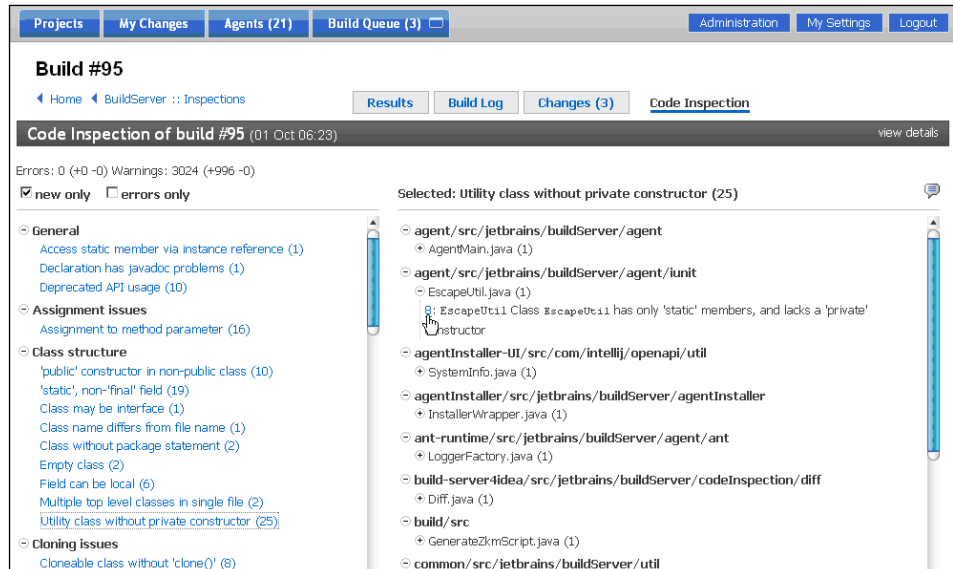
Build runner: **Inspection** Runner for gathering IntelliJ IDEA inspections results.

Path to .ipr file: \*

Inspections profile path:

Build runner options: Ant, **Inspection**, Ipr, MSBuild, Maven2, NAnt, sln2003, sln2005

### Configuring a build



Projects | My Changes | Agents (21) | Build Queue (3) | Administration | My Settings | Logout

**Build #95**

Home > BuildServer :: Inspections | Results | Build Log | Changes (3) | **Code Inspection**

Code Inspection of build #95 (01 Oct 06:23) [view details](#)

Errors: 0 (+0 -0) Warnings: 3024 (+996 -0)

new only  errors only

**Selected: Utility class without private constructor (25)**

- General
  - Access static member via instance reference (1)
  - Declaration has javadoc problems (1)
  - Deprecated API usage (10)
- Assignment issues
  - Assignment to method parameter (16)
- Class structure
  - 'public' constructor in non-public class (10)
  - 'static', non-'final' field (19)
  - Class may be interface (1)
  - Class name differs from file name (1)
  - Class without package statement (2)
  - Empty class (2)
  - Field can be local (6)
  - Multiple top level classes in single file (2)
  - Utility class without private constructor (25)
- Cloning issues
  - Cloneable class without 'clone()' (6)

Files in selected category:

- agent/src/jetbrains/buildServer/agent
  - AgentMain.java (1)
- agent/src/jetbrains/buildServer/agent/iunit
  - EscapeUtil.java (1)
  - EscapeUtil Class EscapeUtil has only 'static' members, and lacks a 'private' constructor
- agentInstaller-UI/src/com/intellij/openapi/util
  - SystemInfo.java (1)
- agentInstaller/src/jetbrains/buildServer/agentInstaller
  - InstallerWrapper.java (1)
- ant-runtime/src/jetbrains/buildServer/agent/ant
  - LoggerFactory.java (1)
- build-server-idea/src/jetbrains/buildServer/codeInspection/diff
  - Diff.java (1)
- build/src
  - GenerateZkmScript.java (1)
- common/src/jetbrains/buildServer/util

### Code inspection results in the web interface

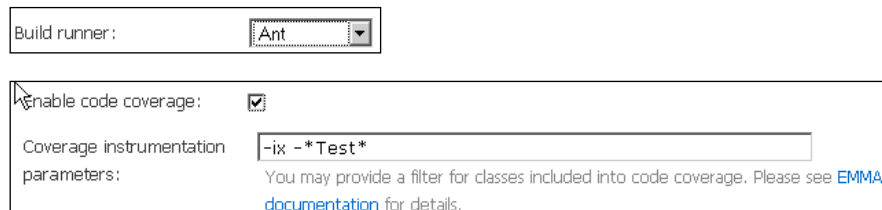
You can find details on setting up the configuration in the online Getting Started Guide.

## Code Coverage Analysis

As you know, this analysis reveals the extent to which a project's source code is covered by tests. In version 1.0, code coverage analysis is available for projects using Ant or IntelliJ IDEA (Ipr) as the build runner.

### Enabling Code Coverage

Enabling code coverage analysis is a simple matter of checking a check box in any Build Configuration using **Ant** or **Ipr** as the runner:



Build runner:

Enable code coverage:

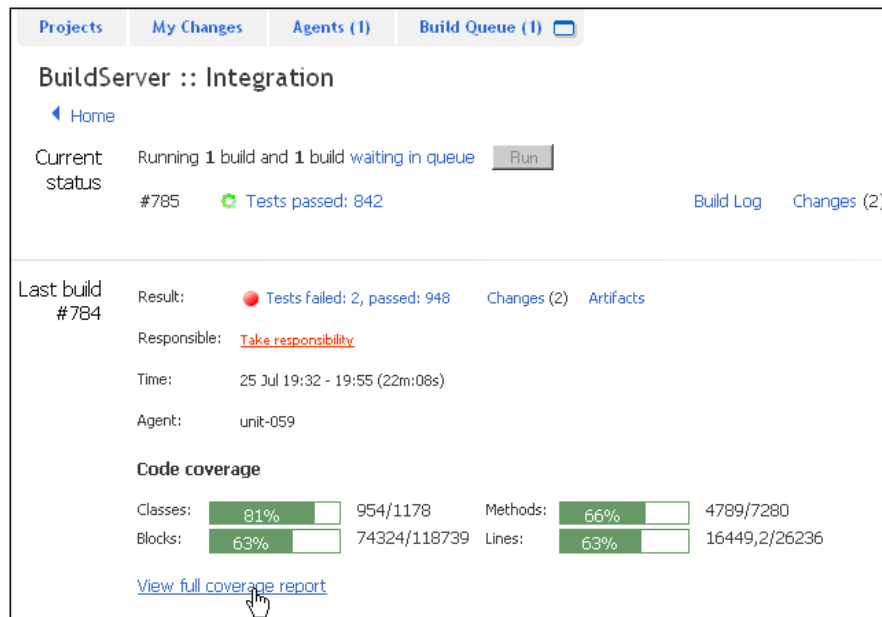
Coverage instrumentation parameters:

You may provide a filter for classes included into code coverage. Please see [EMMA documentation](#) for details.

You can optionally change the instrumentation parameters, but the defaults should be fine for many situations.

### Viewing Results

A summary of the results of the analysis are included on the Build Results page...



Projects | My Changes | Agents (1) | Build Queue (1)

### BuildServer :: Integration

[Home](#)

Current status: Running 1 build and 1 build waiting in queue

#785 ● Tests passed: 842 [Build Log](#) [Changes \(2\)](#)

---

Last build #784

Result: ● Tests failed: 2, passed: 948 [Changes \(2\)](#) [Artifacts](#)

Responsible: [Take responsibility](#)

Time: 25 Jul 19:32 - 19:55 (22m:08s)

Agent: unit-059

**Code coverage**

Classes:	<div style="width: 81%;"><div style="width: 81%;"></div></div> 81%	954/1178	Methods:	<div style="width: 66%;"><div style="width: 66%;"></div></div> 66%	4789/7280
Blocks:	<div style="width: 63%;"><div style="width: 63%;"></div></div> 63%	74324/118739	Lines:	<div style="width: 63%;"><div style="width: 63%;"></div></div> 63%	16449,2/26236

[View full coverage report](#)

...and there is a link that will display a detailed report for those who need it.

## THANK YOU

---

On behalf of the entire JetBrains team, thanks for your interest in reviewing TeamCity 1.0. And once again, if you have questions or need any assistance, don't hesitate to call on us!

### For the TeamCity 1.0 Reviews Program:

**North America:**

Brian Noll

[brain.noll@jetbrains.com](mailto:brain.noll@jetbrains.com)

+1 (609) 714-7883

**Europe:**

Ann Oreshnikova

[editors@jetbrains.com](mailto:editors@jetbrains.com)

+7 812 380 16 41