**What is Jenkins?**

Jenkins is an [award-winning](https://wiki.jenkins-ci.org/display/JENKINS/Awards) application that monitors executions of repeated jobs, such as building a software project or jobs run by cron. Among those things, current Jenkins focuses on the following two jobs:

* **Building/testing software projects continuously**, just like CruiseControl or DamageControl. In a nutshell, Jenkins provides an easy-to-use so-called continuous integration system, making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. The automated, continuous build increases the productivity.
* **Monitoring executions of externally-run jobs**, such as cron jobs and procmail jobs, even those that are run on a remote machine. For example, with cron, all you receive is regular e-mails that capture the output, and it is up to you to look at them diligently and notice when it broke. Jenkins keeps those outputs and makes it easy for you to notice when something is wrong.

**Who is using it?**

[A lot of companies](https://wiki.jenkins-ci.org/pages/viewpage.action?pageId=58001258) and organizations. Most instances tend to run inside the firewall, but Google can tell you [publicly visible instances](http://www.google.com/search?ie=UTF-8&q=%22Dashboard+%5BJenkins%5D%22). We also have some statistics collected from the anonymous usage survey [here](http://jenkins-ci.org/node/282)

**Features**

Jenkins offers the following features:

1. **Easy installation**: Just java -jar jenkins.war, or deploy it in a servlet container. No additional install, no database.
2. **Easy configuration**: Jenkins can be configured entirely from its friendly web GUI with extensive on-the-fly error checks and inline help. There's no need to tweak XML manually anymore, although if you'd like to do so, you can do that, too.
3. **Change set support**: Jenkins can generate a list of changes made into the build from Subversion/CVS. This is also done in a fairly efficient fashion, to reduce the load on the repository.
4. **Permanent links**: Jenkins gives you clean readable URLs for most of its pages, including some permalinks like "latest build"/"latest successful build", so that they can be easily linked from elsewhere.
5. **RSS/E-mail/IM Integration**: Monitor build results by RSS or e-mail to get real-time notifications on failures.
6. **After-the-fact tagging**: Builds can be tagged long after builds are completed.
7. **JUnit/TestNG test reporting**: JUnit test reports can be tabulated, summarized, and displayed with history information, such as when it started breaking, etc. History trend is plotted into a graph.
8. **Distributed builds**: Jenkins can distribute build/test loads to multiple computers. This lets you get the most out of those idle workstations sitting beneath developers' desks.
9. **File fingerprinting**: Jenkins can keep track of which build produced which jars, and which build is using which version of jars, and so on. This works even for jars that are produced outside Jenkins, and is ideal for projects to track dependency.
10. **Plugin Support**: Jenkins can be extended via 3rd party plugins. You can write plugins to make Jenkins support tools/processes that your team uses.

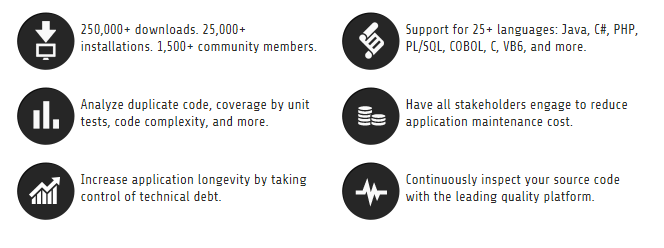
**Jenkins Best Practices**

Continuous Integration with automated test execution has seen broad adoption in recent years. The ideas behind Continuous Integration have changed how companies look at Build Management, Release Management, Deployment Automation, and Test Orchestration. This section provides a set of best practices for Jenkins - A Continuous Integration Solution to provide executives, business managers, software developers and architects a better sense of the development progress and code quality of projects throughout the development lifecycle. (View [Jenkins Best Practices](https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+Best+Practices))

Sonar:

# Continuous Code Quality Management

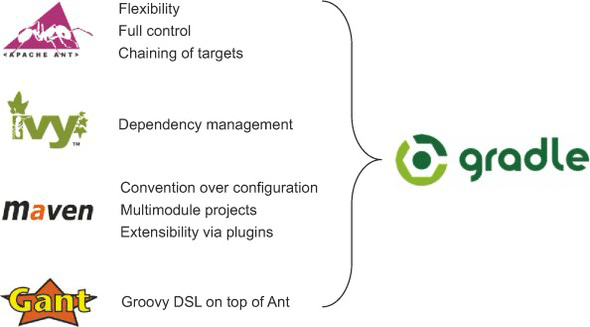
## Improve software quality to increase the efficiency of your development teams and the longevity of your application with our unique Continuous Inspection approach.

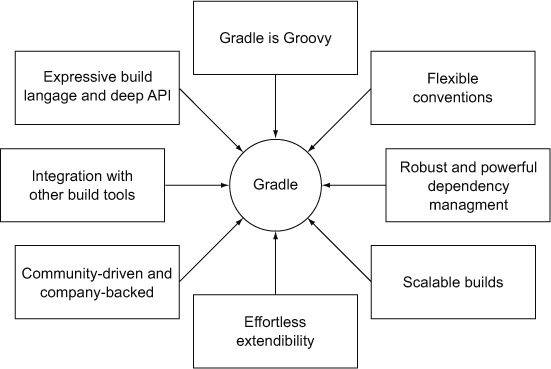


Successful organizations are moving quality analysis up front in the process and making it iterative as part of the overall software development life cycle. Development and test worlds and business approaches as well are evolving toward agile models — continuous value generation/continuous improvement/continuous inspection are essential strategies to improve quality.

Gradle

1. **Gradle** is a project automation tool that **builds** upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the more traditional XML form of declaring the project configuration.

  
**Figure 1: Gradle combines the best features from other build tools.**

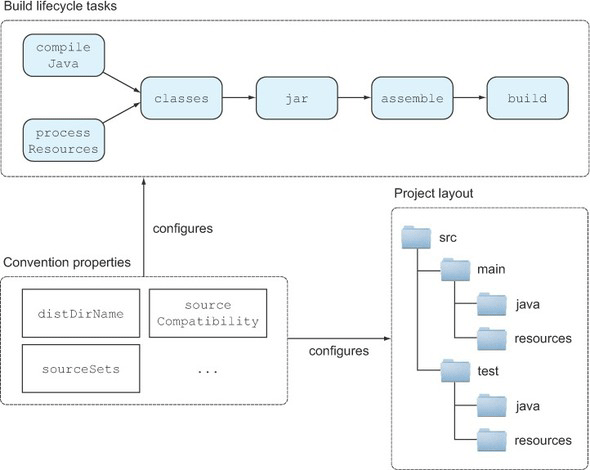
  
**Figure 3: Gradle's compelling feature set.**

### Gradle is Groovy

Prominent build tools like Ant and Maven define their build logic through XML. As we all know, XML is easy to read and write, but can become a maintenance nightmare if used in large quantities. XML isn't very expressive. It makes it hard to define complex custom logic. Gradle takes a different approach. Under the hood, Gradle's DSL is written with Groovy providing syntactic sugar on top of Java. The result is a readable and expressive build language. All your scripts are written in Groovy as well. Being able to use a programming language to express your build needs is a major plus. You don't need to be a Groovy expert to get started. Because Groovy is written on top of Java, you can migrate gradually by trying out its language features. You could even write your custom logic in plain Java — Gradle couldn't care less. Groovy veterans will assure you that using Groovy instead of Java will boost your productivity significantly. A great reference guide is the book *Groovy in Action, Second Edition* by Dirk Koenig et al. (Manning, 2009).

### Flexible Conventions

One of Gradle's big ideas is to give you guidelines and sensible defaults for your projects. Every Java project in Gradle knows exactly where source and test class file are supposed to live, and how to compile your code, run unit tests, generate Javadoc reports, and create a distribution of your code. All of these tasks are fully integrated into the build lifecycle. If you stick to the convention, there's only minimal configuration effort on your part. In fact, your build script is a one-liner. Seriously! Figure 5 illustrates how Gradle introduces conventions and lifecycle tasks for Java projects.

  
**Figure 5: In Gradle, Java projects are build by convention with sensible defaults. Changing the defaults is easy and achieved through convention properties.**

Default tasks are provided that make sense in the context of a Java project. For e

1. **FitNesse** is a web server, a wiki, and an automated testing **tool** for software. It is based on Ward Cunningham's **Framework** for Integrated Test. **FitNesse** is designed to support acceptance testing rather than unit testing in that it facilitates detailed readable description of system function.

# What is FitNesse?

### FitNesse is a software development collaboration tool

Great software requires collaboration and communication. FitNesse is a tool for enhancing collaboration in software development.  
  
FitNesse enables customers, testers, and programmers to **learn what their software should do**, and to automatically compare that to **what it actually does do**. It compares customers' expectations to actual results.  
  
It's an invaluable way to collaborate on complicated problems (and get them right) early in development.  
  
(*The above description is adapted from James Shore's description of Ward Cunningham's*[*Fit Framework*](http://www.fitnesse.org/FitNesse.UserGuide.WritingAcceptanceTests.FitFramework)*, which is one of the test systems upon which FitNesse depends.*)

### FitNesse is a software testing tool.

From another perspective, FitNesse is a lightweight, open-source framework that makes it easy for software teams to:

* Collaboratively define [Acceptance Tests](http://www.fitnesse.org/FitNesse.UserGuide.AcceptanceTests), web pages containing simple tables of inputs and expected outputs.
* Run those tests and see the results (see [Two Minute Example](http://www.fitnesse.org/FitNesse.UserGuide.TwoMinuteExample)).

### FitNesse is a wiki.

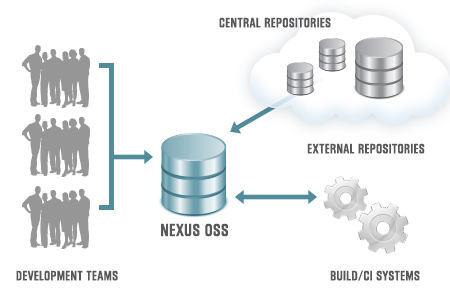
* You can *easily* create and edit pages. (See [Editing Pages](http://www.fitnesse.org/FitNesse.FullReferenceGuide.UserGuide.FitNesseWiki.EditingFitNessePages).)

### FitNesse is a web server.

* It requires **no configuration or setup**.
* Just run it and then direct your browser to the machine where it is running (see [Downloading and Installing](http://fitnesse.org/FitNesseDownload)).

# Nexus Repository Managers

http://www.sonatype.com/assets/title-shadow-6cc4749df8b8500721e3da404f425f1e.png



## Why Nexus?

A repository manager stores and organizes binary software components for use in development, deployment, and provisioning. Repository managers serve five primary purposess:

* Provides a central point for management and governance for all binary software components and their dependencies
* Acts as highly configurable proxy between your organization and public repositories
* Establishes a deployment destination for internally developed binary components
* Enables staging of releases to a temporary repository which can then be used to test and certify a set of artifacts, before they are released
* Supports improved security and privacy surrounding what components are being used across the development organization

Repository managers have become essential infrastructure for component-based software development. To the right is a diagram that depicts where a repository manager fits into a typical development process.

### The Benefits of a Repository Manager

A repository manager encourages collaboration, speeds build times, and improves visibility and control over component usage. Some of the specific benefits you’ll see from a repository manager include:

* **Faster and more reliable builds**

Caching components locally eliminates the need to download dependencies over the Internet, saving time and eliminating the risk that the Internet or remote server won’t be available during the build.

* **Improved collaboration**

It provides a mechanism for developers to share binary components for internally developed software projects. The repository manager becomes a deployment target when components are created, and the standard source of both internally developed and open source components for use in development.

* **Component usage visibility**

By examining the components present in the repository manager you’ll have visibility into what internally developed components are available and what components developers have downloaded from public repositories.

* **Controlled sharing with partners**

Partner-specific repositories with access controls allow you to choose which components and projects to share.

* **Ideal repo for robust governance**

It provides the basis for storing components, which is an integral part of an overall Component Lifecycle Management solution.

Reference:

<http://www.sonatype.com/nexus>

<http://www.fitnesse.org/FitNesse.UserGuide.OneMinuteDescription>

<http://www.sonarsource.com/resources/idc-technology-spotlight/>

<https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins>

FRAMEWORKS:

Service based applications - SOA

## Leverage Struts, Spring, Hibernate, and Axis

Mockito, PowerMock & JUnit

|  |  |
| --- | --- |
| **Struts** | **Spring** |
| Struts is a sophisticated framework offering the easy 2 develop, structured view/presentation layer of the MVC applications. Advanced, robust and scalable view framework underpinning reuse and seperation of concerns to certain extent. | Springs is a Lightweight Inversion of Control and Aspect Oriented Container Framework. Every work in the last sentence carry the true purpose of the Spring framework. It is just not a framework to integrate / plug in at the presentation layer. It is much more to that. It is adaptible and easy to run light weight applications, it provides a framework to integrate OR mapping, JDBC etc., Infact Struts can be used as the presentation tier in Spring. |
| Struts features strictly associate with presentation stuff. It offers Tiles to bring in reuse at presentation level. It offers Modules allowing the application presentation to segregate into various modules giving more modularity there by allowing each module to have its own Custom/Default Request Processor. | Spring provides Aspect Oriented programming, it also solves the seperation of concerns at a much bigger level. It allows the programmer to add the features (transactions, security, database connectivity components, logging components) etc., at the declaration level. Spring framework takes the responsibility of supplying the input parameters required for the method contracts at runtime reducing the coupling between various modules by a method called dependency injection / Inversion of Control. |
| Struts is developed with a Front Controller and dispatcher pattern. Where in all the requests go to the ActionServlet thereby routed to the module specific Request Processor which then loads the associated Form Beans, perform validations and then handovers the control to the appropriate Action class with the help of the action mapping specified in Struts-config.xml file. | Spring does not route the request in a specific way like this, rather it allows to you to design in your own way however in allowing to exploit the power of framework, it allows you to use the Aspect Oriented Programming and Inversion of Control in a great way with great deal of declarative programming with the XML. Commons framework can be integrated to leverage the validation in spring framework too. Morethan this, it provides all features like JDBC connectivity, OR Mapping etc., just to develop & run your applications on the top of this. |
| Struts 2 integrates with Dojo AJAX framework closely and provides many complex UI components out-of-box, such as datepicker, tooltips, etc. | Spring is not. |
| Struts 2 has AJAX theme. | Spring has not. |
| Struts 2 tags are stylesheet-driven, making it easier to develop consistent pages. | Spring is not. |
| Struts 2 checkboxes are stateful, and require no special handling. | Spring is not. |
| Struts is mainly a presentation layer framework, like redirecting to a particular page ,doing client side validations etc which otherwise very tedious using jsp and servlets. | Spring is a complete J2EE framework having seven independent layers which can be used all together or few layers integrated with some other framework. Spring provides declarative transaction management and AOP. One layer of spring is for presentation purpose like struts but for only presentation layer, struts is better than spring. |
| If you want a bunch of taglibs that generate form fields and so forth, Struts is probably the better choice. | Whereas Spring is not in this area. |
| Our UI is mostly click-driven and light on data and validation. It seems to me that most people run into difficulties with Struts when they start moving a lot of data from HTTP into the model. | In Spring this problem does not exist. |

12 BENEFITS OF SPRING MVC OVER STRUTS

[NOVEMBER 10, 2006](http://orangeslate.com/2006/11/10/12-benefits-of-spring-mvc-over-struts/) | [LIJIN](http://orangeslate.com/author/lijinjoseji/)

Spring is a powerful Java application framework, used in a wide range of Java applications. It provides enterprise services to Plain Old Java Objects (POJOs). Spring uses dependency injection to achieve simplification and increase testability.

1. [Spring](http://www.technorati.com/tag/Spring) provides a very clean division between controllers, [JavaBean](http://www.technorati.com/tag/Javabean) models, and views.

2. [Spring’s MVC](http://www.technorati.com/tag/Spring+MVC) is very flexible. Unlike Struts, which forces your Action and Form objects into concrete inheritance (thus taking away your single shot at concrete inheritance in Java), [Spring MVC](http://www.technorati.com/tag/Spring+MVC) is entirely based on interfaces. Furthermore, just about every part of the [Spring MVC](http://www.technorati.com/tag/Spring+MVC) framework is configurable via plugging in your own interface. Of course we also provide convenience classes as an implementation option.

3. Spring, like WebWork, provides interceptors as well as controllers, making it easy to factor out behavior common to the handling of many requests.

4. [Spring MVC](http://www.technorati.com/tag/Spring+MVC) is truly view-agnostic. You don’t get pushed to use JSP if you don’t want to; you can use Velocity, [XLST](http://www.technorati.com/tag/XSLT) or other view technologies. If you want to use a custom view mechanism – for example, your own templating language – you can easily implement the Spring View interface to integrate it.

5. Spring Controllers are configured via IoC like any other objects. This makes them easy to test, and beautifully integrated with other objects managed by Spring.

6. Spring MVC web tiers are typically easier to test than Struts web tiers, due to the avoidance of forced concrete inheritance and explicit dependence of controllers on the dispatcher servlet.

7. The web tier becomes a thin layer on top of a business object layer. This encourages good practice. Struts and other dedicated web frameworks leave you on your own in implementing your business objects; Spring provides an integrated framework for all tiers of your application.

8. No ActionForms. Bind directly to domain objects

9. More testable code (validation has no dependency on Servlet API)

10. Struts imposes dependencies on your Controllers (they must extend a [Struts](http://www.technorati.com/tag/Struts) class), Spring doesn’t force you to do this although there are convenience Controller implementations that you can choose to extend.

11. Spring has a well defined interface to business layer

12. Spring offers better integration with view technologies other than [JSP](http://www.technorati.com/tag/JSP) ([Velocity](http://www.technorati.com/tag/Velocity) / XSLT / [FreeMarker](http://www.technorati.com/tag/Freemarker) / XL etc.)

<http://blogs.isostech.com/web-application-development/struts-2-vs-spring-mvc-battle-frameworks/>

<http://www.withoutbook.com/DifferenceBetweenSubjects.php?subId1=27&subId2=28&d=Difference+between+Struts+and+Spring>

<http://orangeslate.com/2006/11/10/12-benefits-of-spring-mvc-over-struts/>