# Building Java Projects with Gradle

**This case study will walks you through using Gradle to build a simple Java project.**

## What you’ll build

You’ll create a simple app and then build it using Gradle.

## What you’ll need

* **About 15 minutes**
* **A favorite text editor or IDE**
* [**JDK 6**](http://www.oracle.com/technetwork/java/javase/downloads/index.html)**or later**

To **skip the basics**, do the following:

* [Download](https://github.com/spring-guides/gs-gradle/archive/master.zip) and unzip the source repository for this guide, or clone it using [Git](https://spring.io/understanding/Git):git clone <https://github.com/spring-guides/gs-gradle.git>
* cd into gs-gradle/initial
* Jump ahead to [Install Gradle](https://spring.io/guides/gs/gradle/#initial).

**When you’re finished**, you can check your results against the code in gs-gradle/complete.

**Set up the project**

First you set up a Java project for Gradle to build. To keep the focus on Gradle, make the project as simple as possible for now.

### Create the directory structure

In a project directory of your choosing, create the following subdirectory structure; for example, with mkdir -p src/main/java/hello on \*nix systems:

**└── src**

**└── main**

**└── java**

**└── hello**

Within the src/main/java/hello directory, you can create any Java classes you want. For simplicity’s sake and for consistency with the rest of this guide, Spring recommends that you create two classes: HelloWorld.java and Greeter.java.

src/main/java/hello/HelloWorld.java

package hello;

public class HelloWorld {

public static void main(String[] args) {

Greeter greeter = new Greeter();

System.out.println(greeter.sayHello());

}

}

src/main/java/hello/Greeter.java

package hello;

public class Greeter {

public String sayHello() {

return "Hello world!";

}

}

## Install Gradle

Now that you have a project that you can build with Gradle, you can install Gradle.

Gradle is downloadable as a zip file at <http://www.gradle.org/downloads>. Only the binaries are required, so look for the link to gradle-version-bin.zip. (You can also choose gradle-version-all.zip to get the sources and documentation as well as the binaries.)

Unzip the file to your computer, and add the bin folder to your path.

To test the Gradle installation, run Gradle from the command-line:

gradle

If all goes well, you see a welcome message:

:help

Welcome to Gradle 2.3.

To run a build, run gradle <task> ...

To see a list of available tasks, run gradle tasks

To see a list of command-line options, run gradle --help

BUILD SUCCESSFUL

Total time: 2.675 secs

You now have Gradle installed.

## Find out what Gradle can do

Now that Gradle is installed, see what it can do. Before you even create a build.gradle file for the project, you can ask it what tasks are available:

gradle tasks

You should see a list of available tasks. Assuming you run Gradle in a folder that doesn’t already have a build.gradle file, you’ll see some very elementary tasks such as this:

:tasks

== All tasks runnable from root project

== Build Setup tasks

setupBuild - Initializes a new Gradle build. [incubating]

wrapper - Generates Gradle wrapper files. [incubating]

== Help tasks

dependencies - Displays all dependencies declared in root project 'gs-gradle'.

dependencyInsight - Displays the insight into a specific dependency in root project 'gs-gradle'.

help - Displays a help message

projects - Displays the sub-projects of root project 'gs-gradle'.

properties - Displays the properties of root project 'gs-gradle'.

tasks - Displays the tasks runnable from root project 'gs-gradle'.

To see all tasks and more detail, run with --all.

BUILD SUCCESSFUL

Total time: 3.077 secs

Even though these tasks are available, they don’t offer much value without a project build configuration. As you flesh out the build.gradle file, some tasks will be more useful. The list of tasks will grow as you add plugins to build.gradle, so you’ll occasionally want to run **tasks**again to see what tasks are available.

Speaking of adding plugins, next you add a plugin that enables basic Java build functionality.

## Build Java code

Starting simple, create a very basic build.gradle file that has only one line in it:

apply plugin: 'java'

This single line in the build configuration brings a significant amount of power. Run **gradle tasks** again, and you see new tasks added to the list, including tasks for building the project, creating JavaDoc, and running tests.

You’ll use the **gradle build** task frequently. This task compiles, tests, and assembles the code into a JAR file. You can run it like this:

gradle build

After a few seconds, "BUILD SUCCESSFUL" indicates that the build has completed.

To see the results of the build effort, take a look in the build folder. Therein you’ll find several directories, including these three notable folders:

* classes. The project’s compiled .class files.
* reports. Reports produced by the build (such as test reports).
* libs. Assembled project libraries (usually JAR and/or WAR files).

The classes folder has .class files that are generated from compiling the Java code. Specifically, you should find HelloWorld.class and Greeter.class.

At this point, the project doesn’t have any library dependencies, so there’s nothing in the**dependency\_cache** folder.

The reports folder should contain a report of running unit tests on the project. Because the project doesn’t yet have any unit tests, that report will be uninteresting.

The libs folder should contain a JAR file that is named after the project’s folder. Further down, you’ll see how you can specify the name of the JAR and its version.

## Declare dependencies

The simple Hello World sample is completely self-contained and does not depend on any additional libraries. Most applications, however, depend on external libraries to handle common and/or complex functionality.

For example, suppose that in addition to saying "Hello World!", you want the application to print the current date and time. You could use the date and time facilities in the native Java libraries, but you can make things more interesting by using the Joda Time libraries.

First, change HelloWorld.java to look like this:

package hello;

import org.joda.time.LocalTime;

public class HelloWorld {

public static void main(String[] args) {

LocalTime currentTime = new LocalTime();

System.out.println("The current local time is: " + currentTime);

Greeter greeter = new Greeter();

System.out.println(greeter.sayHello());

}

}

Here HelloWorld uses Joda Time’s LocalTime class to get and print the current time.

If you ran gradle build to build the project now, the build would fail because you have not declared Joda Time as a compile dependency in the build.

For starters, you need to add a source for 3rd party libraries.

repositories {

mavenCentral()

}

The repositories block indicates that the build should resolve its dependencies from the Maven Central repository. Gradle leans heavily on many conventions and facilities established by the Maven build tool, including the option of using Maven Central as a source of library dependencies.

Now that we’re ready for 3rd party libraries, let’s declare some.

sourceCompatibility = 1.8

targetCompatibility = 1.8

dependencies {

compile "joda-time:joda-time:2.2"

}

With the dependencies block, you declare a single dependency for Joda Time. Specifically, you’re asking for (reading right to left) version 2.2 of the joda-time library, in the joda-time group.

Another thing to note about this dependency is that it is a compile dependency, indicating that it should be available during compile-time (and if you were building a WAR file, included in the /WEB-INF/libs folder of the WAR). Other notable types of dependencies include:

* providedCompile. Required dependencies for compiling the project code, but that will be provided at runtime by a container running the code (for example, the Java Servlet API).
* testCompile. Dependencies used for compiling and running tests, but not required for building or running the project’s runtime code.

Finally, let’s specify the name for our JAR artifact.

jar {

baseName = 'gs-gradle'

version = '0.1.0'

}

The jar block specifies how the JAR file will be named. In this case, it will rendergs-gradle-0.1.0.jar.

Now if you run gradle build, Gradle should resolve the Joda Time dependency from the Maven Central repository and the build will succeed.

## Build your project with Gradle Wrapper

The Gradle Wrapper is the preferred way of starting a Gradle build. It consists of a batch script for Windows and a shell script for OS X and Linux. These scripts allow you to run a Gradle build without requiring that Gradle be installed on your system. To make this possible, add the following block to the bottom of your build.gradle.

task wrapper(type: Wrapper) {

gradleVersion = '2.3'

}

Run the following command to download and initialize the wrapper scripts:

gradle wrapper

After this task completes, you will notice a few new files. The two scripts are in the root of the folder, while the wrapper jar and properties files have been added to a new gradle/wrapperfolder.

└── initial

└── gradlew

└── gradlew.bat

└── gradle

└── wrapper

└── gradle-wrapper.jar

└── gradle-wrapper.properties

The Gradle Wrapper is now available for building your project. Add it to your version control system, and everyone that clones your project can build it just the same. It can be used in the exact same way as an installed version of Gradle. Run the wrapper script to perform the build task, just like you did previously:

./gradlew build

The first time you run the wrapper for a specified version of Gradle, it downloads and caches the Gradle binaries for that version. The Gradle Wrapper files are designed to be committed to source control so that anyone can build the project without having to first install and configure a specific version of Gradle.

At this stage, you will have built your code. You can see the results here:

build

├── classes

│   └── main

│   └── hello

│   ├── Greeter.class

│   └── HelloWorld.class

├── dependency-cache

├── libs

│   └── gs-gradle-0.1.0.jar

└── tmp

└── jar

└── MANIFEST.MF

Included are the two expected class files for Greeter and HelloWorld, as well as a JAR file. Take a quick peek:

$ jar tvf build/libs/gs-gradle-0.1.0.jar

0 Fri May 30 16:02:32 CDT 2014 META-INF/

25 Fri May 30 16:02:32 CDT 2014 META-INF/MANIFEST.MF

0 Fri May 30 16:02:32 CDT 2014 hello/

369 Fri May 30 16:02:32 CDT 2014 hello/Greeter.class

988 Fri May 30 16:02:32 CDT 2014 hello/HelloWorld.class

The class files are bundled up. It’s important to note, that even though you declared joda-time as a dependency, the library isn’t included here. And the JAR file isn’t runnable either.

To make this code runnable, we can use gradle’s application plugin. Add this to yourbuild.gradle file.

apply plugin: 'application'

mainClassName = 'hello.HelloWorld'

Then you can run the app!

$ ./gradlew run

:compileJava UP-TO-DATE

:processResources UP-TO-DATE

:classes UP-TO-DATE

:run

The current local time is: 16:16:20.544

Hello world!

BUILD SUCCESSFUL

Total time: 3.798 secs

To bundle up dependencies requires more thought. For example, if we were building a WAR file, a format commonly associated with packing in 3rd party dependencies, we could use gradle’s [WAR plugin](http://www.gradle.org/docs/current/userguide/war_plugin.html). If you are using Spring Boot and want a runnable JAR file, the [spring-boot-gradle-plugin](http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using-boot-gradle) is quite handy. At this stage, gradle doesn’t know enough about your system to make a choice. But for now, this should be enough to get started using gradle.

To wrap things up for this guide, here is the completed build.gradle file:

build.gradle

apply plugin: 'java'

apply plugin: 'eclipse'

apply plugin: 'application'

mainClassName = 'hello.HelloWorld'

// tag::repositories[]

repositories {

mavenCentral()

}

// end::repositories[]

// tag::jar[]

jar {

baseName = 'gs-gradle'

version = '0.1.0'

}

// end::jar[]

// tag::dependencies[]

sourceCompatibility = 1.8

targetCompatibility = 1.8

dependencies {

compile "joda-time:joda-time:2.2"

}

// end::dependencies[]

// tag::wrapper[]

task wrapper(type: Wrapper) {

gradleVersion = '2.3'

}

// end::wrapper[]

This is one of the **Gradle hands on case study** on how to Build Java projects with Gradle next gen build tool.

Please do n’t circulate this mail outside GS Pool . This is highly confidential for GS academy and GS Account .

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, February 22, 2016 2:18 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Brij Bhushan Ahuja; Ashutosh Yadav ([ashutosh.yadav@igate.com](mailto:ashutosh.yadav@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day9 ( Build Tools:Gradle vs Maven Feature Comparison:22nd Feb )

### Gradle vs Maven Feature Comparison

At long last, a comprehensive feature comparison of Maven vs Gradle that shows in detail what Build Automation requires in the Age of Continuous Delivery

Gradle has been accurately described as the Build Automation system for the Age of Continuous Delivery–and for the first time Gradle Inc. has published a detailed blow-by-blow feature comparison across all the key functional requirements for Build Automation. So if you’re evaluating Gradle for a new pilot, for Enterprise rollout, for a Continuous Delivery (CD) project, or if you’re building a business case to migrate from Maven, this guide can be of help to you.

### Execution Model

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Directed Acyclic Graph |  | For the Maven DAG a goal must depend on one and only one other goal if there is more than one goal associated with a lifecycle phase. Thus the goals of a lifecycle phase form an ordered list. The Maven DAG is only two levels deep. Those constraints prevent many domain scenarios from being properly modelled. Goals must depend on each other even though they have no dependencies. If a goal depends on more than one goal this must be mapped to an ordered list, destroying this relationship. This makes it often impossible to understand relationships from looking at the pom. This is true for humans as well as for the build system itself which for example is not able to decide how to execute goals in parallel. |
| Task Exclusion | [You can exclude any task from being run.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#sec:excluding_tasks_from_the_command_line) | In Maven there is no generic exclude mechanism. Plugins have to implement it themselves. |
| Transitive Exclude | [When you exclude a task, all tasks this task depends on are also automatically excluded if they have no other dependencies.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#sec:excluding_tasks_from_the_command_line) |  |
| Advanced Task Ordering | [Beyond having full control about the dependencies that are created between tasks, Gradle has powerful language constructs to describe execution order between tasks even if tasks depends not on each others output. This can be modelled with shouldRunAfter and mustRunAfter relationships.](https://docs.gradle.org/current/userguide/more_about_tasks.html#sec:ordering_tasks) |  |
| Finalizers | [Tasks can be assigned to finalize another tasks similar to a finalizer clause in Java. They are always run after another task is executed, regardless whether this task fails or not. This is very powerful for example when doing lifecycle management for containers or databases.](https://docs.gradle.org/current/userguide/more_about_tasks.html#sec:the_idea_behind_gradle_tasks) |  |
| Task Dependency Inference | Gradle objects are aware of which tasks produce certain content. For example, the object representing the Java binary directory knows that the compile task produces the binaries. Any task that has the Java binary directory as input will automatically depend on the compile task. It does not need to be declared manually. This makes the build easier to maintain and more robust. |  |
| True Multi Task Execution | [You can specify multiple tasks when executing the build and no task in the resulting DAG is executed twice. This is very helpful when you want to have an ad-hoc combination of behavior or when you deal with a build you can not change (for example, when configuring a CI job for the build). This is not possible with Ant or Maven.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#N10798) | When you execute for example mvn clean compile test, the compile action is executed twice. |
| Continue Execution After Failures | [Does not stop as soon as the first failure is encountered. Executes every task to be executed where all of the dependencies for that task completed without failure. Enables discoery of as many failures as possible in a single build execution with a very nice aggregated error report at the end.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#sec:continue_build_on_failure) | The Maven reactor has a --fail-at-end option. It is less powerful compared to Gradle as it is not task based. It does not build a dependent module if anything went wrong in the module it depends on, even if the library of the module it depends on was properly produced. |
| Dry Run | [Run a build to see which tasks actually get executed without executing the task actions.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#sec:dry_run) |  |
| Profile Report | [Creates a performance profile for a build. Lists summary times and details for both the configuration phase and task execution. The times for configuration and task execution are sorted with the most expensive operations first. The task execution results also indicate if any tasks were skipped (and the reason) or if tasks that were not skipped did no work.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#sec:profiling_build) |  |
| Build User Oriented Output Mode [Tweet](https://twitter.com/share) | The build output is very important part for the build user experience. In most other build tools the default output is geared towards a build author trying to debug a problem. This leads to a very verbose output hiding often important warnings and messages that are actually relevant for a developer running the build. Gradle's default output is geared towards a developer running the build and showing only messages that are relevant in this context and not abusing the log output as a poor man's progress indicator, for example when executing tests. |  |
| Rerouting output from external tools and libraries | [The build output is very important for the build user experience. If you integrate with external tools and libraries their console output might be very verbose. In Gradle System.out and log output of Java Util Logging, Jakarta Commons Logging and Log4j is re-reouted to the Gradle logging system. You can define per external tool you are integrating with to which log level the output should be routed.](https://docs.gradle.org/current/userguide/logging.html#sec:external_tools) |  |
| Creating a custom logging schema | [You can replace much of Gradle's logging UI with your own. You might do this, for example, if you want to customize the UI in some way - to log more or less information, or to change the formatting.](https://docs.gradle.org/current/userguide/logging.html#sec:changing_what_gradle_logs) |  |
| Embedded GUI | [In addition to supporting a traditional command line interface and IDE integration, Gradle offers a graphical user interface. This is a stand alone user interface that can be launched with the --gui option.](https://docs.gradle.org/current/userguide/tutorial_gradle_gui.html) |  |
| Task Groups and Descriptions | Every task can have a description and can be assigned to a group. This information is used for reporting and how the build is represented in the IDE. | Plugins and command can have descriptions. The grouping is the implicit grouping of the lifecycle phases. |
| Partial Transitive Multi-Module Builds | Build only the specified project and the ones it depends on. |  |
| Automatic Validation of Task Inputs [Tweet](https://twitter.com/share) | [Based on the input/output model Gradle automatically validates the configuration values of a task before it is executed.](https://docs.gradle.org/current/userguide/userguide_single.html#incremental_tasks) |  |

### Continuous Mode

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Build automatically when sources change |  | There's no global "watch" or "continuous" mode. Each plugin needs to add their own implementation if they want this functionality. |
| Trigger rules are automatically derived from the task definition |  |  |

### Configuration Model

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Declarative Builds | Declarative elements are elements that trigger the creation of executional elements. For example the Gradle Java plugin is a declarative element that when declared creates Gradle tasks to model the full lifecycle of a Java project. Declarative elements are the 'What' of a Gradle build whereas the executional elements are the 'How'. Declarative elements often assume conventions when configuring the executional elements they create. | Maven has one kind of declarative element, the packaging element. By specifying a certain type, e.g. jar, all the goals required to build a Java project are wired with the lifecycle phases. |
| Fine Grained Declarative Elements | Gradle has many fine grained declarative elements such as SourceSets or Android Product Flavors. They are at the heart of the Gradle DSL and make the Gradle build language so much richer then what you can do with Maven. They keep the build concise, easy to use, maintain and understand even if you have complex requirements. They provide a balance between structure and hiding unnecessary flexibility and enough flexibility so your developers can describe their requirements | Maven has no fine grained declarative elements. This is one major cause of the extreme inflexibility of Maven. Instead of providing you a rich, deeply integrated model you get a shallow, simplistic, overly rigid framework that is not able to deal with unanticipated requirements. |
| Custom Declarative Elements | In Gradle, every plugin can contribute their own coarse or fine grained declarative elements. This enables you to provide a declarative approach even for your custom domains. It also allows every technology that integrates with Gradle to be modelled as a first class citizen and make it a pleasure to be used. |  |
| Domain Objects Container | [Every domain object describing your build, be it repositories, source directories, plugins or dependencies are stored in a responsive container that you can register listener with. You can write for example a company wide Gradle plugin that add rules to the repository container that no build script is allowed to add a repository that is not using a white list of URL's. You have full control over what particular builds scripts add to a build. You can react in any way. Augment or modify what has been added, let the build fail or issue a warning. You can add define dependencies that are only added for example if a build adds a particular plugin. A very, very powerful feature.](https://docs.gradle.org/current/javadoc/org/gradle/api/NamedDomainObjectContainer.html) |  |
| Customizable Declarative Rules | Sometimes the default rules how declarative elements create executional elements don't tell the story you want. For example when declaring sources you might want a completely different toolchain than the one that is by default defined in Gradle and represented by the tasks Gradle creates as a result of the declaration. In the latest version of Gradle you can now change those rules. So far this is only available in the C/C++ and nightly Android support. It will we provide to Java later this year. |  |
| Multiple Mix-Ins for Pre-Configuring Builds | In Gradle you can mix-in as many objects as you want to pre-configure your build via Gradle plugins. It is important to understand that Gradle plugins are conceptually very different from Maven plugins. | In Maven you can define one and only object (Parent POM) that pre-configures your actual project. This is a severe limitation to organize your build logic comprehensible and with DRY. Being able to mix-in multiple pom's into another pom is long-planned feature in the Maven world but it has not been released yet. |
| Fine Grained Build Event Listener | [Gradle allows you to hook into every part of the build configuration and execution lifecycle for injecting custom behaviour, extracting information, adding additional logging and a tons of other use cases.](https://docs.gradle.org/current/userguide/build_lifecycle.html#build_lifecycle_events) |  |
| User Based Behavior Injection | [You can put custom listeners into your Gradle user home that hook into every Gradle build that is executed on your machine. With the lifecycle listeners described above you can add whatever custom behavior you want to individualize your build experience. For example adding and configuring the Gradle announcement plugin that pops up a window when the build is finishing or failing or adding a special repository that your are just using personally.](https://docs.gradle.org/current/userguide/init_scripts.html) |  |
| Per Build Behavior Injection | [Similar to user based behavior injection you can also specify on the command line additional listeners that hook into a build. This can be very helpful for example if you want your CI build to have specific behavior (e.g. fail if a non-standard repository is used).](https://docs.gradle.org/current/userguide/init_scripts.html) |  |
| Custom Distributions | [Every Gradle distribution has an init.d directory in which you can put custom scripts that pre-configure your build environment. You can use this to apply company wide custom rules that are enforced across all builds of all teams, to provide build-in set up tasks for developers, and so much more. Together with the Gradle wrapper you can easily distribute those custom distributions.](https://docs.gradle.org/current/userguide/init_scripts.html) |  |
| Dynamic Task Creation | [Sometimes you want to have a task whose behavior depends on a large or infinite number value range of parameters. A very nice and expressive way to provide such tasks are task rules.](https://docs.gradle.org/current/userguide/more_about_tasks.html#N10F07) |  |
| Custom Build Script Names | In Gradle you can change the name of the build scripts. For example in multi-module builds the build scripts of a subproject Foo can be named foo.gradle. That way you don't loose orientation when you have many subproject build scripts open up in an editor. |  |

### Dependency Management

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Transitive Dependencies | [One of the main benefits of using a dependency management system is managing transitive dependencies. Gradle takes care of downloading and managing transitive dependencies.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:exclude_transitive_dependencies) | One reason Maven 2 got popular is because of the support of transitive dependenciy management, a concept that was introduced into the Java world by Ivy in 2004. |
| 3rd Party Dependency Cache | [Dependencies from remote repositories are downloaded and cached locally. Subsequent builds use cached artifacts to avoid unnecessary network traffic.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:dependency_cache) |  |
| Concurrency Safe Cache | [Gradle's local dependency cache was designed from the ground up with concurrent saftey in mind. Whether it's utilizing Gradle's built-in parallel build support or running multiple Gradle builds at once, you can be sure your builds will remain fast and reliable.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:cache_locking) | With Maven you can’t run multiple builds at the same time against the same instance of the cache. There is an external Maven extension that provides this functionality. We don’t know how production ready this is and why it is not in the core, as it has been around a while. |
| Repository Aware Cache | [Repository metadata is kept alongside cached dependencies. This allows Gradle to effectively handle cases in which artifacts differ for the same dependency in different repositories. Additionally, Gradle enforces that a cached dependency was retrieved from one of the project's configured repositories. If a dependency isn't available from an "approved" repository, the build will fail.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:cache_repository_independence) | When different projects assigned to different repositories use the same cache they can overwrite each other when using the same dependency coordinates. This can lead to nasty, non-reproducible behavior, specially when dealing with dynamic dependencies. |
| Checksum Based Cache | [Checksums are stored and used to both ensure cache integrity and optimize bandwidth usage. Artifacts are only downloaded if an existing cached version doesn't exist or remote version's checksum is different.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:cache_checksum_storage) |  |
| Sync Cache with Repository | [Gradle has a --refresh-dependencies option to ignore all cached entries for resolved modules and artifacts. A fresh resolve will be performed against all configured repositories, with dynamic versions recalculated, modules refreshed, and artifacts downloaded. However, where possible Gradle will check if the previously downloaded artifacts are valid before downloading again. This is done by comparing published SHA1 values in the repository with the SHA1 values for existing downloaded artifacts.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:cache_refresh) |  |
| Parallel Download of Dependencies |  | When a dependency is not found in the cache it is downloaded from the remote repository. When multiiple dependencies need to be downloads, Maven downloads them in parallel. |
| Reading of of POM Metadata Format | [Gradle is compatible with the POM Metadata format and can retrieve dependencies from any Maven compatible repository.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:maven_repo) |  |
| Reading of IVY Metadata Format | [Gradle is compatible with the Ivy Metadata format. This includes declaring dependencies on particular configurations. Ivy metadata is exposed to custom resolution rules allowing you to filter on artifact branch, status or other custom metadata information.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:ivy_repositories) |  |
| Dynamic Dependencies | [Resolved dependency versions can be dynamic. Gradle supports the Maven snapshot mechanism but is more powerful than that. You can declare a dependency on the latest release, most current development version, or even the latest 4.x build.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:dynamic_versions_and_changing_modules) | Via Snapshot Dependencies |
| Dynamic Dependencies Selection Rules | [Define custom rules to select a specific version when a dynamic dependency is declared. The rules can be based on names and version but also extended metadata like branch or status. The rules can also differ based on the environment the build is happening, e.g. local or CI.](https://docs.gradle.org/current/userguide/dependency_management.html#component_selection_rules) |  |
| Version Conflict Resolution | [By default, Gradle resolves conflicts to the newest requested version. You can customize this behavior.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:dependency_resolution) | Maven is not doing full conflict resolution. It simply always choses the version that has the lowest nesting level in the transitive dependency graph. |
| Central Versioning Definition | [Declare library versions in a central location, avoiding duplication in your build scripts and establishing project conventions.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:custom_versioning_scheme) |  |
| Enforcing Central Versioning | [While Maven allows projects to use dependency versions suggested by their parents, these version can be overridden. Gradle allows stricter governance of a dependency by using substitution rules to optionally force projects to use a particular version of a dependency.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:custom_versioning_scheme) |  |
| Substitution of Compatible Libraries | [Use dependency substitution rules to identify that dependency should be treated as similar. For example log4j and log4j-over-slf4j. Tell Gradle that only one should be selected and use Gradle conflict resolution to pick the newest version from both of them. Similar use cases are situations where you have libraries like spring-all and spring-core in dependency graph. Without properly modelling this the proper behavior of your application depends on the very fragile order in your classpath.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:module_substitution) |  |
| Modeling Releasable Units | [Unique module dependencies are often logically associated to each other. In other words, a collection of dependencies should be treated like a single "releasable unit". Think of a large application framework with many subcomponents. With Gradle, we can ensure we use the same version across dependencies, including transitive dependencies, to maintain reliable runtime compatibility.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:releasable_unit) |  |
| ReplacedBy Rules | [Often projects migrate to new module names as part of a version upgrade. For example, Google's "guava" project was formally known as "google-collections". Declare that dependencies, including transitive dependencies, still referencing the old module name should use the new artifacts. This feature ensures both that we avoid multiple versions of the same library as well as doing conflict resolution across both projects.](https://docs.gradle.org/current/userguide/dependency_management.html#sec:module_replacement) |  |
| Enhanced Metadata Resolution Support | [Dependency metadata can be modified after repository metadata is download but before it is chosen by Gradle as the final resolved version. This allows the creation of custom rules to do things like declare modules as changing (or snapshot) versions, or use a custom status scheme.](https://docs.gradle.org/current/userguide/dependency_management.html#component_metadata_rules) |  |
| Replacement of external and project dependencies | [Dynamically replace external dependencies for project dependencies and vice versa. Especially helpful when only a subset of your modules are checked out locally.](https://docs.gradle.org/current/userguide/dependency_management.html#dependency_substitution_rules) |  |
| Custom Dependency Scopes | [Don't be limited by a predefined set of dependency scopes (compile, runtime, etc). Gradle allows you to define arbitrary dependency scopes. For example for integration tests that you may model in your build, to provision toolchains you need in your build, etc ...](https://docs.gradle.org/current/userguide/dependency_management.html#sub:configurations) |  |
| Custom Repository Layout | [Declare repositories with custom layouts. With custom layouts you can effectively treat nearly any file system directory structure as an artifact repository.](https://docs.gradle.org/current/userguide/dependency_management.html#N154B8) |  |
| File Based Dependencies | [Not all dependencies are available from external repositories. Declare dependencies on filesystem resources when using a managed dependency isn't practical or when migrating legacy builds.](https://docs.gradle.org/current/userguide/dependency_management.html#sub:file_dependencies) |  |
| Variant aware dependency management | The new (still experimental) Gradle software model supports variant aware dependency management. This capability allows to pick the right dependency (including transitive ones) depending on the platform or context you are in. For example the Java version you are using, for the native world those are things like ABI and OS, for Android it would be build types and product flavors. |  |

### Publishing Artifacts

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Generating and Publishing POM Metadata |  |  |
| Generating and Publishing Ivy Metadata |  |  |
| Publishing Multiple Artifacts | Gradle can publish multiple artifacts per project with different metadata. Be it an API and an implementation jar, a library and a test-fixture or variants for different Java platforms. |  |
| Publish Custom Metadata | Gradle can not just use custom metadata for dependency resolution but can of course also publish this custom metadata, like status schemas, branches an artefact origins from, etc ... |  |

### Performance

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Incremental Builds | Gradle checks in between build runs whether the input and output of a task has changed since the last build invocation. If not, the task is considered up to date and is not executed. Gradle also considers the configuration of the task as part of its input. So when you change the source compatibility setting for the compiler task it is no loner considerd up to date for example. The average build time is dramatically reduced for many builds by this feature. |  |
| API for Incremental Subtasks | [When Gradle discovers that the input or output of a task has changed between build runs, the task is executed again. The task can use the incremental API to learn what files exactly have changed. With this information the task may not need to rebuild everything.](https://docs.gradle.org/current/userguide/custom_tasks.html#incremental_tasks) |  |
| Compiler Daemon | When you need to fork the compilation process, Gradle creates a daemon process that is reused within a multi project build. This provides a dramatic speed improvement for the compilation process. |  |
| Parallel Subproject Builds | In multi-module builds separate subprojects can be built in parallel. |  |
| Parallel Task Execution | Gradle ships already with a non-exposed switch to do parallel task execution. It will be enabled by default for Gradle builds using the still experimental new configuration model. For current Gradle build we do not recommend to use it as there is a chance for race conditions and irreproducible behavior as Gradle can not currently prevent, that build configuration is changed during execution time. |  |
| Parallel Scheduler for Subtask Parallelization | Sometimes a task can be broken into subtasks that run in parallel, so that it is faster overall. Gradle divides compilation steps for native languages (Assembler/C/C++/Obj-C/Obj-C++) into subtasks and submits them to a shared resource pool. It is important that parallelization within a task can synchronize with the parallelization of the overall build, so that resources are used efficiently. Gradle has an internal API for this currently, but we plan to make this public later this year. |  |

### Build Infrastructure Administration

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Self Provisioning Build Environment | With the Gradle wrapper the Gradle build environment is auto-provisioned. Furthermore you can determine which version should be used to build your project. | This is not supported by Maven core. There is an external Maven extension that has copied the Gradle Wrapper functionality. It is missing features like checksum support. |
| Version Controlled Build Environment Configuration | Important parameters for configuring the build environment can be stored in version as part of your project. No need for the developers to set them up manually. This includes the Gradle version to be used, the configuration for the JVM running the build and the JDK to be used for running the build. |  |
| Custom Distributions | [Every Gradle distribution has an init.d directory in which you can put custom scripts that pre-configure your build environment. You can use this to apply custom rules that are enforced, to provide build-in set up tasks for developers, and so much more. Together with the Gradle wrapper you can easily distribute those custom distributions.](https://docs.gradle.org/current/userguide/init_scripts.html) |  |

### Embedding

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| SDK for embedded usage | Provided by the Gradle Tooling API |  |
| Version agnostic | The Gradle tooling API is back and forward compatible. With a particular version of the Tooling API you can drive build across all Gradle version since 1.0. |  |
| Querying for Project model | [You can query Gradle for the details of a build, including the project hierarchy and the project dependencies, external dependencies (including source and Javadoc jars), source directories and tasks of each project.](https://docs.gradle.org/current/userguide/userguide_single.html#sec:embedding_introduction) |  |
| Query for Build environment information | Gradle provides programmatically access to information about the build environment. This includes information about the Gradle Version, The Gradle User Home directory and the Java Home directory. |  |
| Execute a build | You can execute a build and listen to stdout and stderr logging and progress (e.g. the stuff shown in the 'status bar' when you run on the command line). |  |
| Build Operation Cancellation | All operations initiated via the Gradle Tooling API are gracefully cancellable at any time during the build |  |
| Support custom JVM settings | JVM args provided via the Gradle Tooling API take precedence over gradle.properties |  |
| Provide Eclipse Project Model | The Gradle Tooling API provides a model of how your project is mapped to an Eclipse project. |  |
| Provide IDEA Project Model | The Gradle Tooling API provides a model how to your project is mapped to the IDEA model. |  |
| Provide support for custom Project Model | You can write a Gradle plugin to add custom metadata for the Gradel Tooling API. You can use this for example when you integrate your own product/customizations with Gradle. |  |
| Run specific tests | The TestLauncher API allows running specific JUnit or TestNG Tests within one or more Test Tasks |  |
| Register for progress events | Register for events of task or test progress to get informed about the process of the build with rich information about the processed task and test |  |
| Run Continuous Builds | Gradle Tooling API provides the functionality to run a build programmatically in "continuous build" mode. Changes on build inputs (File changes) continuously trigger specified tasks to run. |  |

### Ant Integration

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Run Targets from Ant Build |  |  |
| Embed Ant Tasks | [You can deeply integrate any default, optional or cutom Ant task.](https://docs.gradle.org/current/userguide/ant.html#N11365) |  |
| Deep Import of Ant Builds | [In Gradle you can import an Ant Build at runtime. All the Ant targets are wrapped around Gradle tasks and thus fully integrated. You can for example enable incremental and parallel builds for them and other Gradle tasks can depend on them. It is also possible to read configuration information from your Ant build or inject configuration from Gradle into your Ant build.](https://docs.gradle.org/current/userguide/ant.html#N11405) |  |
| Partial Replacement of Ant Targets with Gradle Tasks | [You can even make your Ant targets dependent on Gradle tasks. This allows you to fully replace some part of your Ant build towards Gradle and import the other part.](https://docs.gradle.org/current/userguide/ant.html#N11405) |  |

### Android Support

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Support for Building Android Applications |  |  |
| Maintained by the Google Android Team | The Google Android team is the owner and maintainer of the Gradle Android Plugin. |  |
| Support for official Android Project Layout and Conventions |  |  |
| Android Domain Build Language | Describe your project via an Android-specific DSL. Tell Gradle \_about\_ your project, now how to build it. Avoid the overly verbose need to explicitly implement the complex build process of Android projects. |  |
| Full Integration with Android Studio | Gradle is deeply integrated with Android Studio, the official Android IDE. In fact, Android Studio has no internal builder, it instead delegates all build tasks to Gradle. This "unified build system" ensures correctness across all your builds, whether they be run from Android Studio, the command line, or a continuous integration build server. |  |
| Build Variants | Declare your build types and product flavors and let Gradle handle the rest. No need to copy and paste build logic over and over for each of your project's variants. Use the expressive DSL to declare only what is different about your variants. |  |
| Android Library Projects | Android library projects are treated as first-class citizens in Gradle's multi-project build model. This enables all the benefits provided by Gradle's multi-project build support, such as project dependencies, and incremental builds. | No build type of product flavor support. |
| Manifest Merging | Manifests and resource files are automatically merged, and individual entries overridden from variant source files. Build APKs with different application names, SKD versions or even required permissions. | Since the Maven Android plugin lacks support for modeling build variants, manifest and resource merging must be configured explicitly. |
| Per-Variant Dependency Management | Gradle's advanced dependency management features are available to Android projects and can be configured uniquely for each build variant. Declare dependencies relevant to only certain variants or customize dependency resolution for each of your build variants. |  |
| Application Signing | Automate signing your applications. Debug build variants are, by default, signed with a debug key for installation on development devices. Declare additional signing configurations for publication to the Google Play store. |  |
| ProGuard Support | Easily configure ProGuard on your project to enable obfuscation and minification of the built APK. Configure ProGuard independently for each build type, allow an unoptimized APK for development and optimized one for release. |  |
| Unit Testing | Run unit tests against your Android application or library. Unit tests run in a standard Java JVM against a mocked Android SDK implementation facilitating fast-feedback test development, eliminating the overhead of building a separate APK and running tests of a device or emulator. |  |
| On-device Functional Testing | For tests that require a real Android environment for proper execution, bundle your test in a separate APK to be installed and run on an Android device or emulator. Separate APKs are built for each build variant allowing you to build and test every variant of your app in a single Gradle build. | Test code must live in a separate project causing unnecessary workspace bloat. |
| APK Splits | Efficiently build multiple APKs for differing device display densities or ABIs by configuring Gradle to reuse shared build outputs. |  |
| Multidex Support | Utilize multidex support to avoid the 65k method limit imposed by Android DEX files. |  |
| NDK Support | Integrate your build with the Android NDK. Build application that depend on native libraries, projects that compile C/C++ into native libraries or both. |  |
| Databinding | Using binding classes generated by your Gradle build you can drastically simplify the way you interact with Android views. No more interacting with views directly, simply update the backing object and your view is kept up to date. |  |

### Native Languages – C/C++/Assembler Support

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Build C/C++/Obj-C/Obj-C++/Assembler | Gradle has built in support for compiling and linking programs using Assembler, C/C++ and Obj-C/C++. Gradle can build shared and static libraries and executables. | ?   There is native support for Maven via the Maven NAR plugin. We have not evaluated it in depth and therefore can't compare it yet with Gradle's native support. A contribution that does this comparison would be much appreciated. |
| Model variants of a native component | Easily model variants of a native component like support for different ABI's, OS, build types, etc. | ? |
| Supports GCC | [Gradle supports building with GCC4 on Linux, Windows (with Cygwin and MingW) and Mac OS X.](https://docs.gradle.org/current/userguide/nativeBinaries.html#native-binaries:tool-chain-support) | ? |
| Supports Clang | [Gradle supports building with Clang on Linux and Mac OS X.](https://docs.gradle.org/current/userguide/nativeBinaries.html#native-binaries:tool-chain-support) | ? |
| Supports MS Visual C++ | [Gradle supports building with Microsoft's Visual C++ compiler on Windows. (VS 2010 and VS 2013 supported)](https://docs.gradle.org/current/userguide/nativeBinaries.html#native-binaries:tool-chain-support) | ? |
| Generates Windows Resources | [Gradle uses Microsoft's resource compiler to build Windows resource script files into your application.](https://docs.gradle.org/current/userguide/nativeBinaries.html#native_binaries:windows-resources) | ? |
| Parallel Compilation | When building native code, Gradle divides the compilation step into parallelizable tasks and executes them in a shared resource pool. This speeds up the single project case and ensures that Gradle does not consume too many resources in parallel multi-project builds. | ? |
| Precompiled Headers | [Gradle makes it easy to use precompiled headers when building your software. Precompiled headers can speed up compilation times if your project has many header files that are included in most of your source code. Precompiled headers is a compiler-specific optimization to cache an already parsed version of your headers.](https://docs.gradle.org/current/userguide/nativeBinaries.html#native_binaries:preCompiledHeaders) | ? |
| Build Shared/Static Libraries, Executables |  | ? |
| Build mixed language binaries | Gradle can build separate languages (e.g., Assembler and C) and link them into a single executable or library. | ? |
| CUnit Test Support | Gradle supports testing C applications with CUnit. | ? |
| GoogleTest Support | Gradle supports testing C++ applications with GoogleTest. | ? |
| Makefile support | Gradle does not come with built-in support for extracting a build from a Makefile, but Gradle's Exec task can be used to wrap and existing Makefile when migrating to Gradle. | ? |
| Build JNI Libraries | Gradle does not come with an out-of-the-box recipe for building a JNI library, but you can use a custom task to generate the headers and build a Shared Library as usual. | ? |
| Dependency Management | You can use the current Gradle dependency management support to support binary sharing. But it is not fully tailored yet for the needs of native domain. Soon our new variant aware dependency management will provide the first full solution for dependency management in the native world. | ? |

### Miscellaneous

|  |  |  |
| --- | --- | --- |
| **FEATURE** | << OLE Object: Picture (Device Independent Bitmap) >> **GRADLE 2.5** | << OLE Object: Picture (Device Independent Bitmap) >> **MAVEN 3.3.3** |
| Task Name Abbreviation when executing | [You can call a build action by abbreviations. For example if you type 'gradle uA' it will call the task uploadArchives if there are not naming ambiguities.](https://docs.gradle.org/current/userguide/tutorial_gradle_command_line.html#N1080F) |  |
| Project Name Abbreviation when executing | You can reference a project by abbreviations. For example if you type ‘gradle :ser-fun-t:build’ it will call the task build of the project server-functional-test if there are no naming ambiguities. |  |
| Build Comparison | [You can run a build against two different versions of Gradle and Gradle will generate a report that compares the build output of boths. This is useful when migration to a newer version of Gradle.](https://docs.gradle.org/current/userguide/comparing_builds.html) |  |
| Fast Release Cycles And High Activity | Since the release of Gradle 1.0 in June 2012, there have been 18 minor or major releases. Gradle releases a new version usually every 6-8 weeks. Most minor releases in Gradle add a significant set of new features. See for example: <https://docs.gradle.org/1.7/release-notes> | Since the release of Maven 2 in October 2005 there have been only 5 major or minor releases. The minor releases have been mostly bug fix releases with hardly any new functionality. See for example:<https://issues.apache.org/jira/secure/ReleaseNote.jspa?projectId=12316922&version=12330161> |

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, February 17, 2016 12:14 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Brij Bhushan Ahuja; Ashutosh Yadav (ashutosh.yadav@igate.com)  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 8 ( Tools:JIRA :Hands-on:17th Feb )  
**Importance:** High

<< File: JIRA\_Step\_by\_steps\_Handson\_GS\_Academy\_17\_02\_16\_Puru.pdf >>

Forget to attach the file . Please find the attachment for the same.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, February 17, 2016 11:36 AM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Brij Bhushan Ahuja; Ashutosh Yadav (ashutosh.yadav@igate.com)  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 8 ( Tools:JIRA :Hands-on:17th Feb )  
**Importance:** High

Hi Team,

Please go thu above attached JIRA hands on case study document . Try to finish it by Friday EOD.

**Below are the list of topics that have covered in this hands on study material:**

1- JIRA Scheme

2- JIRA Issues and Issue types

• 2.1- What is JIRA Issue?

• 2.2- Issue Types

3- JIRA Components

4- JIRA screen

5- Issue Attributes

6- Issue Security Schemes

• 6.1- System Administration

7- How to create an issue in JIRA

• 7.1- Sub-Task

• 7.2- WorkFlows

• 7.3- Plug-ins in JIRA

• 7.4- JIRA Agile

• 7.5- Creating issue in Agile

• 7.6- How to create an Epic in Agile

• 7.7- Use of Clone and Link in JIRA

8- Reports in JIRA

• 8.1- Kanban Board and Managing issues

8.2- JIRA Scrum Vs JIRA Kanban.

Please don’t circulate these materials about side GS pool . It is highly confidential for GS Academy and GS Account.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 16, 2016 5:00 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Brij Bhushan Ahuja; Ashutosh Yadav ([ashutosh.yadav@igate.com](mailto:ashutosh.yadav@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 8 (Build Tool:Maven :Hands-on:16th Feb )  
**Importance:** High

+Ashutosh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 16, 2016 3:52 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Brij Bhushan Ahuja  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 8 (Build Tool:Maven :Hands-on:16th Feb )  
**Importance:** High

<< File: Maven\_Case\_Study\_Hands\_On\_Approch\_Puru\_GS\_Academy\_16\_02\_16.pdf >>

Hi Team,

Please go thu above attached Maven hands on case study document . Try to finish it by EOD tomorrow .

It consist of 4 studies step by step (it is fully hands-on)namely :

1. How to convert existing java project to maven project
2. How to create simple java project using maven in eclipse
3. How to create dynamic web project using maven in eclipse
4. How to install maven plugin(m2eclipse) in eclipse

If you are not able to do in the office . Please do it @home by installing the latest version of Apache Maven.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Thursday, February 11, 2016 12:47 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 7 (Topics Covered detailed Status)

Hi Team ,

While you are preparing for GS client interviews you need to cover below mentioned topics in details in 1- 2 weeks times with hands on approach . Most of the topics we have covered over the time recently .

Cloud you please provide us the status on the topics that you have finished and practice from you end . According we will start the mock interview for the coming GS client interviews.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Main Stream** | **Concepts** | **Status** | **Covered** | **Pending** |
|  | **ER Design, Design Patterns** |  |  |  |
|  | **UML/OOAD-Class, Sequence, Process flow, Deployment Diagram** |  |  |  |
| Design | **Clustering, Load Balancing** |  |  |  |
|  | **Performance Bench marking** |  |  |  |
|  | **Heap dump and Thread dump analysis** |  |  |  |
|  | **Java Profiling** |  |  |  |
|  | **Reconciliations** |  |  |  |
|  | **Fail overs and application backup** |  |  |  |
|  | **Security Attacks and protection** |  |  |  |
|  |  |  |  |  |
|  | Searching-Binary Search | - |  |  |
|  | Sorting-Insertion, Selection, Bubble, Quick, Merge, Heap, Radix. |  |  |  |
|  | Data Structures- Stack, Queue, List, Map, Set, Linked List, **Tree**, Graph, Heap, No-Object linked list |  |  |  |
| Algorithms | **Caching-LRU, LFU, FIFO** |  |  |  |
|  | Puzzles |  |  |  |
|  | MORE… |  |  |  |
|  |  |  |  |  |
|  | Collections |  |  |  |
| Java | **Concurrent Package** |  |  |  |
|  | **Threading** |  |  |  |
|  | Sockets, IO operations |  |  |  |
|  | Features in Java 1.5,1.6, 1.7, 1.8, NIO, Java Memory and GC |  |  |  |
|  | Code Refactoring |  |  |  |
|  |  |  |  |  |
|  | HQL, SQL, SP, Named Query |  |  |  |
|  | Caching- 1st level, 2nd level (Eh cache, etc.,), Query Cache, **Eh cache implementation – Efficient designing, Session management** |  |  |  |
| Hibernate | **Fetching Strategies(Fetch and JOIN), Joins** |  |  |  |
|  | Associations(1-1, 1-many, many-1) |  |  |  |
|  | **Designing applications using associations and Fetching Strategies for improved performance and robustness(ACID)** |  |  |  |
|  | Connection Pooling – Integration with CP vendors |  |  |  |
|  | Hibernate versioning of Objects |  |  |  |
|  | Hibernate Inheritance – 3 ways |  |  |  |
|  | Hibernate API-CRUD operations |  |  |  |
|  |  |  |  |  |
|  | DI, MVC, AOP- Auditing, Method logging |  |  |  |
|  | Resource Bundling- Property file loading, Validations |  |  |  |
| Spring/Othe | **Spring Security, Spring Batch** |  |  |  |
| r Mid. ware | Integration with Struts, Hibernate |  |  |  |
|  | **Integrating crone job, mailing, etc.,** |  |  |  |
|  | **Cache Management, Tx. Management** |  |  |  |
|  |  |  |  |  |
|  | Rest –Documentation, Equipping message types, Artefacts required w.r.to consumer, provider |  |  |  |
| Web Services | **SOAP- Security, Documentation, Artefacts required w.r.to consumer, provider** |  |  |  |
|  | **JAXB, XSD, Bindings, WSDL customization** |  |  |  |
|  | **Service End points** |  |  |  |
|  | MORE..... |  |  |  |
|  |  |  |  |  |
| UNIX | Commands- ls, ps, vi, cat, tar, ftp, etc., |  |  |  |
|  |  |  |  |  |
| SQL | **Tricky Queries, SPs, Views, Cursors, Triggers, Backup and Restore** |  |  |  |
|  |  |  |  |  |
| MOCK | How to drive/plan the application from scratch |  |  |  |
| JUNIT | How to drive/plan the application from scratch |  |  |  |
| Maven | How to drive/plan the application from scratch |  |  |  |
| Gradle | How to drive/plan the application from scratch |  |  |  |
| Jenkins | How to drive/plan the application from scratch |  |  |  |
| JIRA/Bugzilla | How to drive/plan the application from scratch |  |  |  |
| Code Review | How to drive/plan the application from scratch |  |  |  |
| Logging | Log 4j, slf4j, other frameworks |  |  |  |
|  |  |  |  |  |

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, February 10, 2016 5:11 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 6 (Build Tool:Ant vs Maven vs Gradle)  
**Importance:** High

Hi Team ,

Please go thu Ant vs Maven vs Gradle comparative detailed understanding docuemnts as below :

Java Build Tools: Ant vs Maven vs Gradle

In the beginning there was Make as the only build tool available. Later on it was improved with [GNU Make](http://www.gnu.org/software/make/). However, since then our needs increased and, as a result, build tools evolved.

JVM ecosystem is dominated with three build tools:

* Apache [Ant](http://ant.apache.org/) with [Ivy](http://ant.apache.org/ivy/)
* [Maven](http://maven.apache.org/)
* [Gradle](http://www.gradle.org/)

[Ant](http://ant.apache.org/) with [Ivy](http://ant.apache.org/ivy/)

[<< OLE Object: Picture (Device Independent Bitmap) >>](http://a3ab771892fd198a96736e50.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/06/ant.png) Ant was the first among “modern” build tools. In many aspects it is similar to Make. It was released in 2000 and in a short period of time became the most popular build tool for Java projects. It has very low learning curve thus allowing anyone to start using it without any special preparation. It is based on procedural programming idea.

After its initial release, it was improved with the ability to accept plug-ins.

Major drawback was XML as the format to write build scripts. XML, being hierarchical in nature, is not a good fit for procedural programming approach Ant uses. Another problem with Ant is that its XML tends to become unmanageably big when used with all but very small projects.

Later on, as dependency management over the network became a must, Ant adopted [Apache Ivy](http://ant.apache.org/ivy/).

Main benefit of Ant is its control of the build process.

[Maven](http://maven.apache.org/)

[<< OLE Object: Picture (Device Independent Bitmap) >>](http://a3ab771892fd198a96736e50.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/06/maven.jpg) Maven was released in 2004. Its goal was to improve upon some of the problems developers were facing when using Ant.

Maven continues using XML as the format to write build specification. However, structure is diametrically different. While Ant requires developers to write all the commands that lead to the successful execution of some task, Maven relies on conventions and provides the available targets (goals) that can be invoked. As the additional, and probably most important addition, Maven introduced the ability to download dependencies over the network (later on adopted by Ant through Ivy). That in itself revolutionized the way we deliver software.

However, Maven has its own problems. Dependencies management does not handle well conflicts between different versions of the same library (something Ivy is much better at). XML as the build configuration format is strictly structured and highly standardized. Customization of targets (goals) is hard. Since Maven is focused mostly on dependency management, complex, customized build scripts are actually harder to write in Maven than in Ant.

Maven configuration written in XML continuous being big and cumbersome. On bigger projects it can have hundreds of lines of code without actually doing anything “extraordinary”.

Main benefit from Maven is its life-cycle. As long as the project is based on certain standards, with Maven one can pass through the whole life cycle with relative ease. This comes at a cost of flexibility.

In the mean time the interest for DSLs (Domain Specific Languages) continued increasing. The idea is to have languages designed to solve problems belonging to a specific domain. In case of builds, one of the results of applying DSL is Gradle.

[Gradle](http://www.gradle.org/)

[<< OLE Object: Picture (Device Independent Bitmap) >>](http://a3ab771892fd198a96736e50.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/06/gradle.png) Gradle combines good parts of both tools and builds on top of them with DSL and other improvements. It has Ant’s power and flexibility with Maven’s life-cycle and ease of use. The end result is a tool that was released in 2012 and gained a lot of attention in a short period of time. For example, Google adopted Gradle as the default build tool for the Android OS.

Gradle does not use XML. Instead, it had its own DSL based on [Groovy](http://groovy.codehaus.org/) (one of JVM languages). As a result, Gradle build scripts tend to be much shorter and clearer than those written for Ant or Maven. The amount of boilerplate code is much smaller with Gradle since its DSL is designed to solve a specific problem: move software through its life cycle, from compilation through static analysis and testing until packaging and deployment.

It is using [Apache Ivy](http://ant.apache.org/ivy/) for JAR dependencies.

Gradle effort can be summed as “convention is good and so is flexibility”.

Code examples

We’ll create build scripts that will compile, perform static analysis, run unit tests and, finally, create JAR files. We’ll do those operations in all three frameworks (Ant, Maven and Gradle) and compare the syntax. By comparing the code for each task we’ll be able to get a better understanding of the differences and make an informed decision regarding the choice of the build tool.

First things first. If you’ll do the examples from this article by yourself, you’ll need [Ant](http://ant.apache.org/), [Ivy](http://ant.apache.org/ivy/), [Maven](http://maven.apache.org/) and [Gradle](http://www.gradle.org/) installed. Please follow installation instructions provided by makers of those tools. You can choose not to run examples by yourself and skip the installation altogether. Code snippets should be enough to give you the basic idea of how each of the tools work.

Code repository <https://github.com/vfarcic/JavaBuildTools> contains the java code (two simple classes with corresponding tests), checkstyle configuration and Ant, Ivy, Maven and Gradle configuration files.

Let’s start with Ant and Ivy.

Ant with Ivy

Ivy dependencies need to be specified in the ivy.xml file. Our example is fairly simple and requires only JUnit and Hamcrest dependencies.

[[ivy.xml](https://github.com/vfarcic/JavaBuildTools/blob/master/ivy.xml)]

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 1 | <ivy-module version="2.0"> |
| 2 | <info organisation="org.apache" module="java-build-tools"/> |

|  |  |
| --- | --- |
| 3 | <dependencies> |
| 4 | <dependency org="junit" name="junit" rev="4.11"/> |

|  |  |
| --- | --- |
| 5 | <dependency org="org.hamcrest" name="hamcrest-all" rev="1.3"/> |
| 6 | </dependencies> |

|  |  |
| --- | --- |
| 7 | </ivy-module> |

Now we’ll create our Ant build script. Its task will be only to compile a JAR file. The end result is the following build.xml.

[[build.xml](https://github.com/vfarcic/JavaBuildTools/blob/master/build.xml)]

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 01 | <project xmlns:ivy="antlib:org.apache.ivy.ant" name="java-build-tools" default="jar"> |
| 02 |  |

|  |  |
| --- | --- |
| 03 | <property name="src.dir" value="src"/> |
| 04 | <property name="build.dir" value="build"/> |

|  |  |
| --- | --- |
| 05 | <property name="classes.dir" value="${build.dir}/classes"/> |
| 06 | <property name="jar.dir" value="${build.dir}/jar"/> |

|  |  |
| --- | --- |
| 07 | <property name="lib.dir" value="lib" /> |
| 08 | <path id="lib.path.id"> |

|  |  |
| --- | --- |
| 09 | <fileset dir="${lib.dir}" /> |
| 10 | </path> |

|  |  |
| --- | --- |
| 11 |  |
| 12 | <target name="resolve"> |

|  |  |
| --- | --- |
| 13 | <ivy:retrieve /> |
| 14 | </target> |

|  |  |
| --- | --- |
| 15 |  |
| 16 | <target name="clean"> |

|  |  |
| --- | --- |
| 17 | <delete dir="${build.dir}"/> |
| 18 | </target> |

|  |  |
| --- | --- |
| 19 |  |
| 20 | <target name="compile" depends="resolve"> |

|  |  |
| --- | --- |
| 21 | <mkdir dir="${classes.dir}"/> |
| 22 | <javac srcdir="${src.dir}" destdir="${classes.dir}" classpathref="lib.path.id"/> |

|  |  |
| --- | --- |
| 23 | </target> |
| 24 |  |

|  |  |
| --- | --- |
| 25 | <target name="jar" depends="compile"> |
| 26 | <mkdir dir="${jar.dir}"/> |

|  |  |
| --- | --- |
| 27 | <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}"/> |
| 28 | </target> |

|  |  |
| --- | --- |
| 29 |  |
| 30 | </project> |

First we specify several properties. From there on it is one task after another. We use Ivy to resolve dependencies, clean, compile and, finally, create the JAR file. That is quite a lot of configuration for a task that almost every Java project needs to perform.

To run the Ant task that creates the JAR file, execute following.

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 1 | ant jar |

Let’s see how would Maven does the same set of tasks.

Maven

[[pom.xml](https://github.com/vfarcic/JavaBuildTools/blob/master/pom.xml)]

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 01 | <project xmlns="<http://maven.apache.org/POM/4.0.0>" |
| 02 | xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>" |

|  |  |
| --- | --- |
| 03 | xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> |
| 04 |  |

|  |  |
| --- | --- |
| 05 | <http://maven.apache.org/maven-v4_0_0.xsd>"> |
| 06 |  |

|  |  |
| --- | --- |
| 07 | <modelVersion>4.0.0</modelVersion> |
| 08 | <groupId>com.technologyconversations</groupId> |

|  |  |
| --- | --- |
| 09 | <artifactId>java-build-tools</artifactId> |
| 10 | <packaging>jar</packaging> |

|  |  |
| --- | --- |
| 11 | <version>1.0</version> |
| 12 |  |

|  |  |
| --- | --- |
| 13 | <dependencies> |
| 14 | <dependency> |

|  |  |
| --- | --- |
| 15 | <groupId>junit</groupId> |
| 16 | <artifactId>junit</artifactId> |

|  |  |
| --- | --- |
| 17 | <version>4.11</version> |
| 18 | </dependency> |

|  |  |
| --- | --- |
| 19 | <dependency> |
| 20 | <groupId>org.hamcrest</groupId> |

|  |  |
| --- | --- |
| 21 | <artifactId>hamcrest-all</artifactId> |
| 22 | <version>1.3</version> |

|  |  |
| --- | --- |
| 23 | </dependency> |
| 24 | </dependencies> |

|  |  |
| --- | --- |
| 25 |  |
| 26 | <build> |

|  |  |
| --- | --- |
| 27 | <plugins> |
| 28 | <plugin> |

|  |  |
| --- | --- |
| 29 | <groupId>org.apache.maven.plugins</groupId> |
| 30 | <artifactId>maven-compiler-plugin</artifactId> |

|  |  |
| --- | --- |
| 31 | <version>2.3.2</version> |
| 32 | </plugin> |

|  |  |
| --- | --- |
| 33 | </plugins> |
| 34 | </build> |

|  |  |
| --- | --- |
| 35 |  |
| 36 | </project> |

To run the Maven goal that creates the JAR file, execute following.

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 1 | mvn package |

The major difference is that with Maven we don’t need to specify what should be done. We’re not creating tasks but setting the parameters (what are the dependencies, what plugins to use…). This shows the major difference between Ant and Maven. Later promotes the usage of conventions and provides goals (targets) out-of-the-box. Both Ant and Maven XML files tend to grow big with time. To illustrate that, we’ll add Maven CheckStyle, FindBugs and PMD plugins that will take care of static analysis. All three are fairly standard tools used, in one form or another, in many Java projects. We want all static analysis to be executed as part of a single target **verify** together with unit tests. Moreover, we should specify the path to the custom checkstyle configuration and make sure that it fails on error. Additional Maven code is following:

[[pom.xml](https://github.com/vfarcic/JavaBuildTools/blob/master/pom.xml)]

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 01 | <plugin> |
| 02 | <groupId>org.apache.maven.plugins</groupId> |

|  |  |
| --- | --- |
| 03 | <artifactId>maven-checkstyle-plugin</artifactId> |
| 04 | <version>2.12.1</version> |

|  |  |
| --- | --- |
| 05 | <executions> |
| 06 | <execution> |

|  |  |
| --- | --- |
| 07 | <configuration> |
| 08 | <configLocation>config/checkstyle/checkstyle.xml</configLocation> |

|  |  |
| --- | --- |
| 09 | <consoleOutput>true</consoleOutput> |
| 10 | <failsOnError>true</failsOnError> |

|  |  |
| --- | --- |
| 11 | </configuration> |
| 12 | <goals> |

|  |  |
| --- | --- |
| 13 | <goal>check</goal> |
| 14 | </goals> |

|  |  |
| --- | --- |
| 15 | </execution> |
| 16 | </executions> |

|  |  |
| --- | --- |
| 17 | </plugin> |
| 18 | <plugin> |

|  |  |
| --- | --- |
| 19 | <groupId>org.codehaus.mojo</groupId> |
| 20 | <artifactId>findbugs-maven-plugin</artifactId> |

|  |  |
| --- | --- |
| 21 | <version>2.5.4</version> |
| 22 | <executions> |

|  |  |
| --- | --- |
| 23 | <execution> |
| 24 | <goals> |

|  |  |
| --- | --- |
| 25 | <goal>check</goal> |
| 26 | </goals> |

|  |  |
| --- | --- |
| 27 | </execution> |
| 28 | </executions> |

|  |  |
| --- | --- |
| 29 | </plugin> |
| 30 | <plugin> |

|  |  |
| --- | --- |
| 31 | <groupId>org.apache.maven.plugins</groupId> |
| 32 | <artifactId>maven-pmd-plugin</artifactId> |

|  |  |
| --- | --- |
| 33 | <version>3.1</version> |
| 34 | <executions> |

|  |  |
| --- | --- |
| 35 | <execution> |
| 36 | <goals> |

|  |  |
| --- | --- |
| 37 | <goal>check</goal> |
| 38 | </goals> |

|  |  |
| --- | --- |
| 39 | </execution> |
| 40 | </executions> |

|  |  |
| --- | --- |
| 41 | </plugin> |

To run the Maven goal that runs both unit tests and static analysis with CheckStyle, FindBugs and PMD, execute following.

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 1 | mvn verify |

We had to write a lot of XML that does some very basic and commonly used set of tasks. On real projects with a lot more dependencies and tasks, Maven pom.xml files can easily reach hundreds or even thousands of lines of XML.

Here’s how the same looks in Gradle.

Gradle

[[build.gradle](https://github.com/vfarcic/JavaBuildTools/blob/master/build.gradle)]

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 01 | apply plugin: 'java' |
| 02 | apply plugin: 'checkstyle' |

|  |  |
| --- | --- |
| 03 | apply plugin: 'findbugs' |
| 04 | apply plugin: 'pmd' |

|  |  |
| --- | --- |
| 05 |  |
| 06 | version = '1.0' |

|  |  |
| --- | --- |
| 07 |  |
| 08 | repositories { |

|  |  |
| --- | --- |
| 09 | mavenCentral() |
| 10 | } |

|  |  |
| --- | --- |
| 11 |  |
| 12 | dependencies { |

|  |  |
| --- | --- |
| 13 | testCompile group: 'junit', name: 'junit', version: '4.11' |
| 14 | testCompile group: 'org.hamcrest', name: 'hamcrest-all', version: '1.3' |

|  |  |
| --- | --- |
| 15 | } |

Not only that the Gradle code is much shorter and, to those familiar with Gradle, easier to understand than Maven, but it actually introduces many useful tasks not covered with the Maven code we just wrote. To get the list of all tasks that Gradle can run with the current configuration, please execute the following.

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 1 | gradle tasks --all |

Clarity, complexity and the learning curve

For newcomers, Ant is the clearest tool of all. Just by reading the configuration XML one can understand what it does. However, writing Ant tasks easily gets very complex. Maven and, specially, Gradle have a lot of tasks already available out-of-the-box or through plugins. For example, by seeing the following line it is probably not clear to those not initiated into mysteries of Gradle what tasks will be unlocked for us to use.

[build.gradle]

[view source](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#viewSource)[print](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#printSource)[?](http://www.javacodegeeks.com/2014/06/java-build-tools-ant-vs-maven-vs-gradle.html#about)

|  |  |
| --- | --- |
| 1 | apply plugin: 'java' |

This simple line of code adds 20+ tasks waiting for us to use.

Ant’s readability and Maven’s simplicity are, in my opinion, false arguments that apply only during the short initial Gradle learning curve. Once one is used to the Gradle DSL, its syntax is shorter and easier to understand than those employed by Ant or Maven. Moreover, only Gradle offers both conventions and creation of commands. While Maven can be extended with Ant tasks, it is tedious and not very productive. Gradle with Groovy brings it to the next level.

Please prepare very well on these topics before the Client interview.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, February 10, 2016 2:03 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 6 (Snake and Ladder Problem)  
**Importance:** High

# Hi Team ,

# Yesterday GS has asked couple of designing API for the given problem statement in the IMD Controllers area. One of them was Snake and Ladder Problems . if they ask you such kind of problems below motions would be the approach .

# Snake and Ladder Problem

Given a snake and ladder board, find the minimum number of dice throws required to reach the destination or last cell from source or 1st cell. Basically, the player has total control over outcome of dice throw and wants to find out minimum number of throws required to reach last cell.

If the player reaches a cell which is base of a ladder, the player has to climb up that ladder and if reaches a cell is mouth of the snake, has to go down to the tail of snake without a dice throw.

[<< OLE Object: Picture (Device Independent Bitmap) >>](http://d1gjlxt8vb0knt.cloudfront.net/wp-content/uploads/snakesladders.jpg)

For example consider the board shown on right side, the minimum number of dice throws required to reach cell 30 from cell 1 is 3.

Following are steps.

a) First throw two on dice to reach cell number 3 and then ladder to reach 22  
b) Then throw 6 to reach 28.  
c) Finally through 2 to reach 30.

There can be other solutions as well like (2, 2, 6), (2, 4, 4), (2, 3, 5).. etc.

**Pseudo code** :  
Print rules of the game  
Makes the board for the player  
Prompt user to start the game  
If (input==’y’)  
Then startGame  
Else  
Leave command prompt  
  
If startGame  
  
Set starting position for user and computer to 1  
  
User throws the dice then computer throws the dice one after other  
  
Prints the final position after each dice throw for the user as well as computer repeatedly  
  
If user reaches the ladder or snake bitten then reflect the final position after such propositions both  
for user and computer  
  
Whoever first reaches the 100 , will win the game , with n predefined attempts  
  
If no one wins with in predefined attempts then prompt user to play the game again or not  
  
Repeat step 1  
  
  
**Demo :**

[<< OLE Object: Picture (Device Independent Bitmap) >>](http://2.bp.blogspot.com/-5LdFENvUH7k/UXleUnmfgwI/AAAAAAAAACU/ws2dTu-mZBQ/s1600/snakeladder.bmp)

Code is here :

**import** **java.io.\***; // used to allow user to input data

**public** **class** **Snakes\_and\_Ladders**

{// start class

//Main Method

**public** **static** **void** **main** (String [] args) **throws** IOException

{// start main method

//Welcoming Screen

BufferedReader myInput2 = **new** BufferedReader (**new** InputStreamReader (System.in));

// Print the welcome screen and instructions

System.out.println ("\t\t\t\t\t\tWelcome To Snakes And Ladders\n\n");

System.out.println ("\t\t\t\t\t\t\tInstructions:");

System.out.println ("\t\t\t This program will simulate a regular snakes and ladders game, where you will be");

System.out.println ("\t\t\t facing the computer. You and the computer start at square 1, and the first one to");

System.out.println ("\t\t\t square 100 wins, however, there will be preset squares which will be the snakes or ladders.");

System.out.println ("\t\t\t Once you land on top of a snake you go down a few squares, and move up if you land");

System.out.println ("\t\t\t\t\t\t on the bottom of a ladder. Good Luck and Have FUN!!!");

**int** counter= **100**,iteration=-**1**; // sets the counter and iteration vaiables to be used in making the board

System.out.println ("-----------------------------------------------------Game Board-----------------------------------------------------------------------------");

/\*

This while loop makes the board for the player to visualize what the

game looks like, it uses a counter to increment or decrement by 1.

It will also subtract by 9 or 10 when necessary to create a board

exactly like the one given.

\*/

**while** (counter > **0**){// start while

**if** (counter%**10** == **0** && counter != **100**){// checks if the counter is at a 90, or 80...

**if**(iteration==-**1**)

{

// subtract 9 from the counter

// and sets it to start adding by one

counter-=**9**;

iteration=**1**;

}

**else**

{

System.out.print(counter+"\t");

counter-=**10**;

iteration=-**1**; // set the counter to start subtract by ones

}

**if**(counter!=**0**)

System.out.print("\n" + counter + "\t"); // just prints out the counter with a line breck

}

**else**

System.out.print(counter + "\t"); // just prints out the counter

counter+=iteration; // sets counter to add by iteration

}// end while

System.out.println();

System.out.println ("----------------------------------------------------------------------------------------------------------------------------------");

String sGame = "y"; // decare variable used to ask user if he wants to play

System.out.print ("Do you want to play? Y or N > "); // ask user if we wants to play the game

sGame = myInput2.readLine (); // reads the user's input into the variable sGame

System.out.print ("\n\n\n\n\n\n");

// While the user says yes, go to startGame method

// startGame is fuction type method, which start the game

**while** (sGame.equals ("y") || sGame.equals ("Y"))

{

sGame = startGame(sGame); // give startGame a variable to work with

}

System.out.println ("\n\n\n\n\t\t\t\t\t\tSEE YA!!");

} //end main method

//-------------------------------------------------------------------startGame Method------------------------------------------------------------------------------

/\*

startGame method:

This method is responsible for organizing the game, asking the user to continue playing,

and setting the important varibales. It will also return a vaule to the main method, which

will reset the game so the user can play again.

\*/

**public** **static** String **startGame** (String start) **throws** IOException // Recieves data from the main method

{// start startGame method

BufferedReader myInput = **new** BufferedReader (**new** InputStreamReader (System.in));

// sets important variables for the game

// NOTE: These variables will change as the game progresses

**int** userPosition = **1**; // sets the default loaction for user's piece to 1

**int** compPosition = **1**; // sets the default loaction for computer's piece to 1

**int** diceRoll = **0**; // creates the first die

**int** diceRoll2 = **0**; // creates the second die

**int** userRoll = **1**; // declares what the user rolled

**int** compRoll = **1**; // declares what the computer rolled

String playAgain = "y"; // sets the play again variable

// declare all the snakes and ladders in a array

**int** snakesLaddersArray [] = **new** **int** [**6**]; // create a 6 element array

// store the snakes and ladders location in the array

snakesLaddersArray [**0**] = **54**;

snakesLaddersArray [**1**] = **90**;

snakesLaddersArray [**2**] = **99**;

snakesLaddersArray [**3**] = **9**;

snakesLaddersArray [**4**] = **40**;

snakesLaddersArray [**5**] = **67**;

**while** (playAgain.equals ("y") || playAgain.equals ("Y")) // loops while the playAgaim vaiable equals "y" or "Y".

{// start playAgain While

// gets the dice roll for user and computer

userRoll = getDice(diceRoll, diceRoll2); // sends data to a function type method called getDice

compRoll = getDice(diceRoll, diceRoll2); // does the same for the computer

System.out.println("---------------------------------------------------------------------------------------------------------------------------");

System.out.println ("\t\t\t\t\t------------------------------------------");

System.out.println ("\t\t\t\t\t|\tYou Rolled a " + userRoll + "\t\t|"); // print the roll the user got

System.out.println ("\t\t\t\t\t|\tThe Computer Rolled a " + compRoll + "\t|"); // print the roll the computer got

System.out.println ("\t\t\t\t\t------------------------------------------");

// hold the user's last position for switching back if current position was greater than 100

userPosition = userPosition + userRoll;

// hold the computer's last position for switching back if current position was greater than 100

compPosition = compPosition + compRoll;

// check to see if user landed on top of a snake or at the bottom of a ladder

// give getP parameters to work with, and recieve a final value which can be printed out

userPosition = getP(userPosition, userRoll, snakesLaddersArray);

// The same goes for compPosition, however compgetP gets an additional

// parameter (userPosition) to check if user has already won

compPosition = compgetP(compPosition, compRoll, snakesLaddersArray, userPosition);

System.out.println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println ("\t\t\t\*\t\tYou are currently on square " + userPosition + "\t\t\t\*"); // print out the user's current location

System.out.println ("\t\t\t\*\t\tThe Computer is currently on square " + compPosition + "\t\t\*"); // print out the computer's current location

System.out.println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

// resets the position values, if the user or the computer won

// so that the user could play the entire game again if he/she wanted to

**if** (userPosition == **100** || compPosition == **100**)

{

userPosition = **1**;

compPosition = **1**;

// asks the user if we wants to play again

System.out.print ("Do you want to play? Y or N > ");

playAgain = myInput.readLine ();

System.out.print ("\n\n\n\n\n\n\n\n\n\n\n\n");

}

**else**

{

// asks the user if we wants to continue playing

System.out.print ("Do you want to play? Y or N > ");

playAgain = myInput.readLine ();

}

}// end playAgain While

**return** playAgain; // returns prameter "playAgain" to main: if the user wants to play the game again

}// end startGame method

//-------------------------------------------------------------------geDice Method------------------------------------------------------------------------------

/\*

getDice method:

This method generates two random numbers bewteen 1 and 6, then

adds them to get a final roll. Next it returns the value to be

diplayed on the screen.

\*/

**public** **static** **int** **getDice** (**int** diceRoll, **int** diceRoll2)

{// start getDies class

diceRoll = (**int**)(Math.random()\***6** )+**1** ; //creates dice roll number 1

diceRoll2 = (**int**)(Math.random()\***6** )+**1** ; //creates dice roll number 2

**int** move = diceRoll + diceRoll2; // adds the two dice rolls to get a final move

**return** move; // return parameter move: this will give the final dice roll back to startGame

}// end getDice class

//-------------------------------------------------------------------getP Method------------------------------------------------------------------------------

/\*

getP method:

This method is responsible for checking if the USER is on

top of a snake, or at the bottom of a ladder, and then

adjusting the user's position to mathch the snake or

ladders value.

\*/

**public** **static** **int** **getP** (**int** userPosition, **int** userRoll, **int** snakesLaddersArray []) **throws** IOException // recieves two parameter from startGame

{// start getP

**if**(userPosition == snakesLaddersArray[**0**]) //if position equals snake 1

{

userPosition = **19**; // set new position

System.out.println ("\t\t\t\t~~~~~~~~~~~~~You Got Bit By A Snake, GO DOWN!!!~~~~~~~~~~~~~");

}

**else** **if** (userPosition == snakesLaddersArray[**1**]) //if position equals snake 2

{

userPosition = **48**; // set new position

System.out.println ("\t\t\t\t~~~~~~~~~~~~~You Got Bit By A Snake, GO DOWN!!!~~~~~~~~~~~~~");

}

**else** **if** (userPosition == snakesLaddersArray[**2**]) //if position equals snake 3

{

userPosition = **77**; // set new position

System.out.println ("\t\t\t\t~~~~~~~~~~~~~You Got Bit By A Snake, GO DOWN!!!~~~~~~~~~~~~~");

}

**else** **if** (userPosition == snakesLaddersArray[**3**]) //if position equals ladder 1

{

userPosition = **34**; // set new position

System.out.println ("\t\t\t\t~~~~~~~~~~~~~You Got A Ladder!! GO UP!!!~~~~~~~~~~~~~");

}

**else** **if** (userPosition == snakesLaddersArray[**4**]) //if position equals ladder 2

{

userPosition = **64**; // set new position

System.out.println ("\t\t\t\t~~~~~~~~~~~~~You Got A Ladder!! GO UP!!!~~~~~~~~~~~~~");

}

**else** **if** (userPosition == snakesLaddersArray[**5**]) //if position equals ladder 3

{

userPosition = **86**; // set new position

System.out.println ("\t\t\t\t~~~~~~~~~~~~~You Got A Ladder!! GO UP!!!~~~~~~~~~~~~~");

}

**if** (userPosition < **0** || userPosition > **112**) // This is ab ERROR TRAP to catch any unwanted system errors that may occur by chance

{

System.out.println ("An error has occured please reset the game!!!!!!");

}

**else** **if** (userPosition > **100**) // checks if user's location if greater then a 100

{

userPosition = userPosition - userRoll; // subtract userRoll from the userposition to get back old position

System.out.println ("OHHH You cant jump, you must land on a 100"); // print out the users current location

}

**else** **if** (userPosition == **100**)

{

System.out.println ("YOU WON, GOOD JOB!!!"); // print out that the user won

}

**return** userPosition; // return the final position to starGame: this position had gone through all checks and test and can be displayed on screen

}// end getP

//-------------------------------------------------------------------compgetP Method------------------------------------------------------------------------------

/\*

compgetP method:

This method is responsible for checking if the COMPUTER is on

top of a snake, or at the bottom of a ladder, and then

adjusting the compuer's position to mathch the snakes or

ladders value.

\*/

**public** **static** **int** **compgetP** (**int** compPosition, **int** compRoll, **int** snakesLaddersArray [], **int** userPosition) **throws** IOException

{// start compgetP

// NOTE: this method is similar to getP, so the comments are the same for both!!

// Look at getP's commments if you need help

**if**(compPosition == snakesLaddersArray[**0**])

{

compPosition = **19**;

System.out.println ("\t\t\t\t~~~~~~~~~~~~~Computer Got Bit By A Snake, HE WENT GO DOWN!!!~~~~~~~~~~~~~");

}

**else** **if** (compPosition == snakesLaddersArray[**1**])

{

compPosition = **48**;

System.out.println ("\t\t\t\t~~~~~~~~~~~~~Computer Got Bit By A Snake, HE WENT GO DOWN!!!~~~~~~~~~~~~~");

}

**else** **if** (compPosition == snakesLaddersArray[**2**])

{

compPosition = **77**;

System.out.println ("\t\t\t\t~~~~~~~~~~~~~Computer Got Bit By A Snake, HE WENT GO DOWN!!!~~~~~~~~~~~~~");

}

**else** **if** (compPosition == snakesLaddersArray[**3**])

{

compPosition = **34**;

System.out.println ("Computer Got TO A Ladder, HE WENT UP!!!");

}

**else** **if** (compPosition == snakesLaddersArray[**4**])

{

compPosition = **64**;

System.out.println ("Computer Got TO A Ladder, HE WENT UP!!!");

}

**else** **if** (compPosition == snakesLaddersArray[**5**])

{

compPosition = **86**;

System.out.println ("Computer Got TO A Ladder, HE WENT UP!!!");

}

**if** (compPosition < **0** || compPosition > **112**) // ERROR TRAP to catch any unwanted system errors that may occur by chance

{

System.out.println ("An error has occured for the computer, please reset the game!!!!!!");

}

**else** **if** (compPosition > **100**) // checks if computers's location if greater then a 100

{

compPosition = compPosition - compRoll; // get old position

System.out.println ("THE COMPUTER CAN'T JUMP!!! He must land on a 100"); // give message that the computer cant jump

}

**else** **if** (compPosition == **100** && userPosition != **100**)

{

System.out.println ("THE COMPUTER WON, YOU FAILED!!!"); // print out that the computer won

}

**return** compPosition; // return the final position to starGame: this position had gone through all checks and test and can be displayed on screen

} // end compgetPy

}//end class

Time complexity of the above solution is O(N) as every cell is added and removed only once from queue. And a typical enqueue or dequeue operation takes O(1) time.

Please try to run it and see the behavior of the code ……………..!!!

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 09, 2016 4:54 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 5 (Java Collection :Imp Questions:Assigemnt )  
**Importance:** High

Hi Team ,

Please try to solve all, the below listed 15 programming question on HashMap collections . Most of the time GS asked such types of question .

Please try to execute these programs in the eclipse ide.

15 Java HashMap Programs And Examples

1) Explain the different ways of creating HashMap in java?

2) How do you add key-value pairs to HashMap?

3) How do you add given key-value pair to HashMap if and only if it is not present in the HashMap?

4) How do you retrieve a value associated with a given key from the HashMap?

5) How do you check whether a particular key/value exist in a HashMap?

6) How do you find out the number of key-value mappings present in a HashMap?

7) How do you remove all key-value pairs from a HashMap? OR How do you clear the HashMap for reuse?

8) How do you retrieve all keys present in a HashMap?

9) How do you retrieve all the values present in a HashMap?

10) How do you retrieve all key-value pairs present in a HashMap?

11) How do you remove a key-value pair from the HashMap?

12) How do you remove a key-value pair from a HashMap if and only if the specified key is currently mapped to given value?

13) How do you replace a value associated with a given key in the HashMap?

14) How do you replace a value associated with the given key if and only if it is currently mapped to given value?

15) How do you get synchronized HashMap in java?

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 09, 2016 3:18 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 5 (SCM:SVN:Hands on )

**Dear all .**

**Please go thu the hands on document on the SVN**

**Step by step subversion on Windows with TortoiseSVN**

* [Introduction](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#Introduction)
* [Subversion installation](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#SubversionInstallation)
* [TortoiseSVN installation](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#SubversionInstallation)
* [Step-by-step setup](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#Step-by-step)
  + Step 0. Global ignore pattern setting (optional)
  + Step 1. Repository (database) creation.
  + Step 2. Initial import - Loading your project in the database
  + Step 3. Creating a working directory - Connecting your project to the database
* [Some remarks](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#remarks)
* [References](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#References)

**Introduction**

Subversion is a multi-platform open source version control system (<http://subversion.tigris.org/>). It consists of a repository database (FSFS or BDB) and several command-line tools. There are GUI front-endsfor the Subversion.

The Subversion is designed for project management with several participants. However it can be used to manage personal projects as well.

The current tutorial is limited to a Windows-based local (without a server) SVN.

**Subversion installation**

Go to the download section of Tigris.org, Open Source Software Engineering Tools ( <http://subversion.tigris.org/> ). The latest command-line binaries and libraries for Windows can be found at this location  <http://www.open.collab.net/downloads/subversion/>. Download the installer. The installation of SVN into the default directory C:\Program Files\Subversion is pretty straightforward.

At this point you can start using SVN. It is not necessary to install any GUI program if you are comfortable with using command-line tools.

**TortoiseSVN installation**

TortoiseSVN is a Subversion client, implemented as a windows shell extension, a plug-in to Windows Explorer (<http://tortoisesvn.tigris.org/>).

The latest version can be downloaded at <http://tortoisesvn.sourceforge.net/downloads>. For my 32-bit processor I used 32 Bit TortoiseSVN-1.3.5.6804-svn-1.3.2.msi Installer. There is another file, TortoiseSVN-1.3.5.6804-svn-1.3.2.md5 in the download area, which is not required for installation. It contains a checksum of the installer file which is useful if you want to verify that you have downloaded the right installer (and that its not corrupted).

You can download my program **HashFile** (<http://www.shokhirev.com/nikolai/programs/progmisc.html>) and verify the installer by checking its MD5 hash (checksum). The checksum must be identical to that found in the TortoiseSVN-1.3.5.6804-svn-1.3.2.md5 file.

This installation is also straightforward, just accept the defaults. The only point of interest is the ASP.Net hack. By default, SVN uses the **.svn** working directories (Linux style for hidden files).

|  |  |
| --- | --- |
| << OLE Object: Picture (Device Independent Bitmap) >> | << OLE Object: Picture (Device Independent Bitmap) >> |
| Default, without the ASP.Net hack | With ASP.Net hack for VS.Net web projects |

This confuses Microsoft ASP.Net and the hack forces SVN to use the  **\_svn** working directories instead. Do not install this feature if you are not going to control your ASP.Net projects by Subversion.

Note, that after installation your Windows Explorer has extra buttons in the main menu

<< OLE Object: Picture (Device Independent Bitmap) >>

and in context (activated by right-clicking) menus.

**Step-by-step setup**

There are various ways of Subversion management (see the manuals). Now we consider the most typical scenario:

* You have an existing project, which you want to control by Subversion
* This is your personal project on a local Windows computer

In this case you should not worry about security and setting up a server.

Suppose your project resides in C:\Projects\MyProject. It contains files and possibly subdirectories:

<< OLE Object: Picture (Device Independent Bitmap) >>

**Step 0 (Optional). Global ignore pattern setting.**You probably do not want to keep track of temporary and some other types of files. Right-click on any folder, and launch TortoiseSVN/Settings:

  << OLE Object: Picture (Device Independent Bitmap) >>

Alternatively you can click on File/TortoiseSVN/Settings in the Windows Explorer main menu.

<< OLE Object: Picture (Device Independent Bitmap) >>

In the General section you can list all ignore file types separated by **spaces** (\*.bak \*.~\* in the above example). Note that this setting affects all working directories. See the section 5.25 of the User Guide for more options.

**Step 1. Repository creation.** On your hard drive create the directory for repositories of all your projects, e.g. C:\SVN. Create inside it a sub-directory \MyProject. Right-click on MyProject and choose TortoiseSVN/Create repository here:

<< OLE Object: Picture (Device Independent Bitmap) >>

 Select the default Native File System (FSFS) option and click the OK button:

<< OLE Object: Picture (Device Independent Bitmap) >>

This converts C:\SVN\MyProject into a repository with the following contents:

<< OLE Object: Picture (Device Independent Bitmap) >>

So far this is an **empty** repository, even though Subversion has created several directories and files! We need to fill it with our project files and connect it with our working project directory. All this required several, sometimes counter-intuitive actions.

**Step 2. Initial import.** Somewhere in your hard drive (e.g. in C:\tmp) create a directory (e.g. \new) with the following three subdirectories:

C:\tmp\new\branches

C:\tmp\new\tags

C:\tmp\new\trunk

The resulting structure is:

<< OLE Object: Picture (Device Independent Bitmap) >>

This structure is necessary for more advanced project management, but it does not hurt to create it beforehand. Some tutorials first recommend to import this structure into the repository and later add the project. I recommend the following shortcut.

* Backup your project (C:\Projects\MyProject.), just in case.
* Delete all unnecessary files that are not included in your global ignore list
* **Move** the contents of \MyProject into the trunk subdirectory (C:\tmp\new\trunk). We will need an empty directory later, anyway.
* Import the 'new' directory into the repository (Right-click/TortoiseSVN/Import):

<< OLE Object: Picture (Device Independent Bitmap) >>

Selelect URL as [file:///C:/SVN/Myproject](file:///C:\SVN\Myproject) (forward slashes!):

<< OLE Object: Picture (Device Independent Bitmap) >>

The "Import finished" message shows C:\tmp\new\\* :

<< OLE Object: Picture (Device Independent Bitmap) >>

Do not worry about the "tmp\new" messages, just check what was actually imported into the repository. Right-click on C:\SVN\MyProject and start TortoiseSVN/Repo-browser:

<< OLE Object: Picture (Device Independent Bitmap) >>

Navigate to file///C:/SVN/MyProject/trunk:

<< OLE Object: Picture (Device Independent Bitmap) >>

Note that the files from the ignore list were not imported. Also there is no traces of  'C:\tmp\new'. It is no longer useful and it can be deleted.

**Step 3. Creating a working directory.** Now you have the repository with all your files and the empty C:\Projects\MyProject (remember, we moved all its contents to C:\tmp\new\trunk?). "To get your hands on blessed, completely approved, and fully loaded Subversion directory, you need to check it out from your repository" [[6](http://www.shokhirev.com/nikolai/programs/SVN/svn.html#Charlie)]. Right-click on C:\Projects\MyProject and choose "**SVN Checkout"**:

<< OLE Object: Picture (Device Independent Bitmap) >>

Set URL to [file:///C:/SVN/MyProject/trunk](file:///C:\SVN\MyProject\trunk) and Checkout directory to C:\Projects\MyProject:

<< OLE Object: Picture (Device Independent Bitmap) >>

Press OK. Checkout has finished:

<< OLE Object: Picture (Device Independent Bitmap) >>

The "blessing" is reflected in the small icons on your project directory and all containing files (and subdirectories).

<< OLE Object: Picture (Device Independent Bitmap) >>

Such a "blessed" directory is called a working directory. Now you can start working with you project and learn more about Subversion.

**Some remarks**

* Sending (checking in) your changes to the repository: Right-click on selected files then "SVN Commit"
* Adding files to the repository. This is a two-step process:
  1. Right-click on selected files then "TortoiseSVN/Add"
  2. Right-click on selected files then "SVN Commit"

On most of the system SVN is installed . So please start doing the hands on the SVN .

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 09, 2016 3:00 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Kundan K Sinha; Ajay S Sastry; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 5 (SCM:SVN)  
**Importance:** High

<< File: subversion\_GS\_Puru.ppt >>

Hi All ,

Please go thu the SVN (SCM:Source code Management ) tool . Please go thu each and every slide . Tomorrow 3 to 4 pm I am planning to have session on SVN and its features .

@Varadha\Gopi : Please book a conf room for the same .

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, February 08, 2016 5:27 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry; Brij Bhushan Ahuja; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Tulsi Gupta; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 4 (Devops and jenkins, ant , maven and Gradle:Case Study)  
**Importance:** High

I am yet to get any response on this . Please share me all the GS interview Questions that you have appeared in the past. Please send me ASAP.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, February 08, 2016 1:49 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry; Brij Bhushan Ahuja; Ajaya K Sahoo; Sameer K Bishi; Sadhu C Paikaray; Tulsi Gupta; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 4 (Devops and jenkins, ant , maven and Gradle:Case Study)  
**Importance:** High

Hi All ,

Request you to share the GS interview question that you have attended in the past. I am in the process of consolidating all the question that recently GS as has asked across the project . Please Send ASAP .Please also circulate this mail if have missed any names in the list.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Thursday, February 04, 2016 11:24 AM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry; Brij Bhushan Ahuja; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com)); Sameer K Bishi; Sadhu C Paikaray ([sadhu.paikaray@igate.com](mailto:sadhu.paikaray@igate.com)); Tulsi Gupta  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 4 (Devops and jenkins, ant , maven and Gradle:Case Study)  
**Importance:** High

Adding Sameer K Bishi ….!!

Sameer ,

Please start going thu all the study materials and assignment one by one . Please try to finish each and every thing as much as possible .

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 02, 2016 6:14 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry; Brij Bhushan Ahuja; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com)); Sadhu C Paikaray ([sadhu.paikaray@igate.com](mailto:sadhu.paikaray@igate.com)); Tulsi Gupta  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 4 (Devops and jenkins, ant , maven and Gradle:Case Study)

+Tulsi

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 02, 2016 6:08 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry; Brij Bhushan Ahuja; Ajaya K Sahoo ([ajaya.sahoo@igate.com](mailto:ajaya.sahoo@igate.com)); Sadhu C Paikaray ([sadhu.paikaray@igate.com](mailto:sadhu.paikaray@igate.com))  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 4 (Devops and jenkins, ant , maven and Gradle:Case Study)  
**Importance:** High

HI All ,

Today we have discussed each and every thing on Jenkins . Now it’s time to have some real hands on Jenkins via case studies . Below mentioned case study will walk you thu all the hands aspect of Jenkins for different level on implementations .

It is suggested to perform this case study on your local systems in coming days(By this week). If want you can use the current on goings projects(like IDATAGen , GSPAM and MAT tool) for this case study also …………..!!!

It is suggested to install Jenkins latest version on each and every system and try to execute this case study without fail . In the hands session have already explain you how to download and install latest version of Jenkins.

I will also share this case study in the GS academy study materials in few day’s time ………!!!

## Continuous integration with Jenkins – Case Study.

This article describes how to build a continuous integration cycle for Java development with the Jenkins continuous integration build server.

**Table of Contents**

[**1. Using the Jenkins build server**](http://www.vogella.com/tutorials/Jenkins/article.html#ci)

[**2. Installation**](http://www.vogella.com/tutorials/Jenkins/article.html#installation)

[**2.1.**](http://www.vogella.com/tutorials/Jenkins/article.html#installation_starting)

[**2.2. Installing of the Jenkins server on Ubuntu**](http://www.vogella.com/tutorials/Jenkins/article.html#installation_ubuntu)

[**2.3. Using the .WAR file of Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#installation_jenkins)

[**3. Configure Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration)

[**3.1. Configuration the JDK location**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration_java)

[**3.2. Secure Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration_user)

[**3.3. Generate ssh key for Jenkins user**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration_ssh)

[**4. Jenkins management**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsmanagement)

[**4.1. Plug-in management**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_pluginmanagement)

[**4.2. Restart your Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_pluginmanagement_restart)

[**5. Support for the Git version control systems**](http://www.vogella.com/tutorials/Jenkins/article.html#scm)

[**6. Setting up a Jenkins job**](http://www.vogella.com/tutorials/Jenkins/article.html#job)

[**7. Prerequisites for build Android apps with Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#androiddescription_prereqs)

[**8. How to build your Android application with Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild)

[**8.1. Overview**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild1)

[**8.2. Android Jenkins build job prerequisites**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild_prerequisites)

[**8.3. Installation of the Android SDK**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild_sdkinstallation)

[**8.4. Jenkins Plug-ins for installation**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild_installation)

[**8.5. Running an Android build with Jenkins**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild_runbuild)

[**8.6. Other useful plug-ins**](http://www.vogella.com/tutorials/Jenkins/article.html#android_jenkinsbuild_tips)

[**9. Jenkins backup and copying files**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_filesystem)

[**9.1. Jenkins backup and copying files**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_filesystem)

[**9.2. Managing Jenkins with Git**](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_git)

[**10. About this website**](http://www.vogella.com/tutorials/Jenkins/article.html#supportandlicense)

[**11. Links and Literature**](http://www.vogella.com/tutorials/Jenkins/article.html#resources)

[**11.1. Jenkins links**](http://www.vogella.com/tutorials/Jenkins/article.html#hudsonlinks)

[**11.2. vogella GmbH training and consulting support**](http://www.vogella.com/tutorials/Jenkins/article.html#resources_general)

## 1. Using the Jenkins build server

Continuous integration is a process in which all development work is integrated as early as possible and the resulting artifacts are automatically created and tested. This process should identify errors as very early in the process.

Jenkins is one open source tool to perform continuous integration and build automation. The basic functionality of Jenkins is to execute a predefined list of steps. The trigger for this execution can be time or event based. For example every 20 minutes or after a new commit in a Git repository.

The list of steps can, for example, include:

* perform a software build with Apache Maven or Gradle
* Run a shell script
* Archive the build result
* Afterwards start the integration tests

Jenkins also monitors the execution of the steps and allows to stop the process if one of the steps fails. Jenkins can also send out notifications about the build success or failure.

Jenkins can be extended by additional plug-ins, e.g., for building and testing Android applications or to support the Git version control system.

## 2. Installation

Jenkins can be started via the command line or can run in a web application server. Under Linux you can also install Jenkins as a system service.

For most platforms you have native packages, see the [**Jenkins Homepage**](http://jenkins-ci.org/).

### 2.2. Installing of the Jenkins server on Ubuntu

Jenkins provides Debian/Ubuntu packages which install Jenkins and register Jenkins as start service. See the [**Install Jenkins on Ubuntu description**](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu).

Jenkins stores all the settings, logs and build artifacts in its home directory. The default installation directory is */var/lib/jenkins* under Ubuntu.

This creates a */etc/init.d/jenkins* start script which starts Jenkins automatically at boot time. If you installed Jenkins locally, you find it running under the following URL: [**http://localhost:8080/**](http://localhost:8080/)

### 2.3. Using the .WAR file of Jenkins

Download the *jenkins.war* file from [**Jenkins Homepage**](http://jenkins-ci.org/).

You can also start Jenkins directly via the command line with java -jar jenkins\*.war. If you start it locally, you find it running under the following URL: [**http://localhost:8080/**](http://localhost:8080/)

To run it in your Tomcat server, put the .WAR file into the *webapps* directory. If you start Tomcat, your Jenkins installation will be available under [**http://localhost:8080/jenkins**](http://localhost:8080/jenkins)

**Note**

If the jenkins.war is deployed in your *webapps* directory, but cannot be started and the tomcat manager says “﻿FAIL - Application at context path /jenkins could not be started ”, you may need to grant the permissons for ﻿JENKINS\_HOME.

﻿cd /usr/share/tomcat7

﻿sudo mkdir .jenkins

﻿sudo

chown tomcat7:nogroup .jenkins

This makes the .jenkins folder writable and Jenkins can use it.

## 3. Configure Jenkins

### 3.1. Configuration the JDK location

Before using Jenkins to build Java applications, you need to configure the location or it where your JDK installation is. Select Manage Jenkins and afterwards Configure System.

<< OLE Object: Picture (Device Independent Bitmap) >>

Enter the correct path to your JDK, Apache Ant and Maven and press the Save button below. Jenkins can also install these for your automatically.

<< OLE Object: Picture (Device Independent Bitmap) >>

### 3.2. Secure Jenkins

It is recommended to secure Jenkins. Manage Jenkins and then Configure Global Security. Select the Enable security flag. The easiest way is to use Jenkins own user database. Create at least the user "Anonymous" with read access. Also create entries for the users you want to add in the next step.

<< OLE Object: Picture (Device Independent Bitmap) >>

On the login page, select Create an account to create the users you just gave access.

<< OLE Object: Picture (Device Independent Bitmap) >>

<< OLE Object: Picture (Device Independent Bitmap) >>

Go to Manage Jenkins, Manage and Assign Roles and then Assign Roles to grant the newly created user additional access rights.

<< OLE Object: Picture (Device Independent Bitmap) >>

Navigate to Manage Roles to define access restrictions in detail. Pattern is a regex value of the job name. The following grants unregistered users read-only access to your build jobs that start with the C-MASTER or M-MASTER prefix and only those.

<< OLE Object: Picture (Device Independent Bitmap) >>

### 3.3. Generate ssh key for Jenkins user

If you want to access a private Git repo, for example at Github, you need to generate an ssh key-pair. Create a SSH key with the following command: sudo -u jenkins ssh-keygen.

## 4. Jenkins management

### 4.1. Plug-in management

Jenkins can be extended via additional plug-ins with more functionality. You can configure your plug-ins via the Manage Jenkins → Manager Plugins link.

To install plugins in Jenkins select use the Manage Jenkins → Manager Plugins link and search for the plugin you want to install. Select it from the list and select to install it and restart Jenkins.

The following table is a summary of commonly used plug-ins.

**Table 1. Jenkins plug-ins**

| **Plug-in name** | **Description** | **URL** |
| --- | --- | --- |
| Git Plugin | This plugin allows use of Git as a build SCM. | <https://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin> |
| Xvnc plugin | This plugin allows projects to run xvnc during a build. This allows for example to run tests which requires a display to run on a virtual display. To use this plug-in you need to connect once to your vncserver on the command line to provide a password. Use for example the following commands.  # install vncserver  apt-get install vnc4server  # switch to jenkins user  sudo su jenkins  # connect to vncserver which creates the password  vncserver :10 | wiki.jenkins-ci.org/display/JENKINS/Xvnc+Plugin |
| Gradle Plugin | This plugin allows to run Gradle builds, e.g., as required for Android, via Jenkins. | <https://wiki.jenkins-ci.org/display/JENKINS/Gradle+Plugin> |
| Maven Plugin | This plugin allows to run Maven builds. | <https://wiki.jenkins-ci.org/display/JENKINS/Maven+Project+Plugin> |
| GitHub plugin | This plugin integrates Jenkins with Github projects. | <https://wiki.jenkins-ci.org/display/JENKINS/Github+Plugin> |
| Publish Over SSH Plugin | This plugin allows to publish build artifacts via ssh | <https://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin> |
| Github Pull Request Builder | This plugin allows to build Github Pull Requests | <https://wiki.jenkins-ci.org/display/JENKINS/GitHub+pull+request+builder+plugin> |

### 4.2. Restart your Jenkins

You can manually restart Jenkins by adding *restart* as URL parameter.

<< OLE Object: Picture (Device Independent Bitmap) >>

## 5. Support for the Git version control systems

Jenkins supports the Git version control system via a plugin. Select the Manage Jenkins → Manager Plugins link. Here you have to install the Git Plugin.

To clone a Git repostory via Jenkins you need to enter the email and user name for your Jenkins system. For this switch into your job directory and run the git config command.

# Need to configure the Git email and user for the Jenkins job

# switch to the job directory

cd /var/lib/jenkins/jobs/Android/workspace

# setup name and email

sudo git config user.name "jenkins"

sudo git config user.email "[test@gmail.com](mailto:test@gmail.com)"

## 6. Setting up a Jenkins job

The build of a project is handled via jobs in Jenkins. Select New Item from the menu

<< OLE Object: Picture (Device Independent Bitmap) >>

Afterwards enter a name for the job and select Freestyle Job. Press OK to create a new Job in Jenkins.

The next page allows you to configure your job. If you for example using Git, enter the URL to the Git repository. If the repository is not public, you may also need to configure the credentials.

<< OLE Object: Picture (Device Independent Bitmap) >>

Specify when and how your build should be triggered. The following example polls the Git repository every 15 min and triggers a build, if something has changed in the repo.

<< OLE Object: Picture (Device Independent Bitmap) >>

I typically delete the workspace before a build to avoid any side-effect. In the Build section you can add a build step, e.g., a Maven build.

<< OLE Object: Picture (Device Independent Bitmap) >>

Press Save to finish the job definition. Press Build Now on the job page to validate the job works as expected.

<< OLE Object: Picture (Device Independent Bitmap) >>

After a while the job should go to green or blue (depending on your configuration), if successful. Click on the job and afterwards on Console Output to see the log file in case of an error or to validate that log looks as expected.

<< OLE Object: Picture (Device Independent Bitmap) >>

## 7. Prerequisites for build Android apps with Jenkins

The following descriptions for build Android applications with Jenkins assume that you have already a correct Gradle build file for your Android application. See [**Android build tutorial**](http://www.vogella.com/tutorials/Android/article.html).

## 8. How to build your Android application with Jenkins

### 8.1. Overview

Jenkins supports the automatic building and testing of Android applications.

### 8.2. Android Jenkins build job prerequisites

To create a build job on Jenkins you need to have a working build setup. The Android tooling generates automatically a valid Gradle build file.

### 8.3. Installation of the Android SDK

The Android SDk must be available on the server. If you build server has no display server you need to install the Android SDK from the command line. You also need to configure the location for the Android SKD for Jenkins.

# download the Android SDK via wget

# for the correct URL see <http://developer.android.com/sdk/index.html>

# and look in the section "SDK Tools Only"

wget <link from the above website>

# example

# wget <http://dl.google.com/android/android-sdk_r24.2-linux.tgz>

# unpack it

tar zxvf filename

# Add the new directory to your patch

# assumes you exported it to /opt/

# this must be set for the Jenkins user

export ANDROID\_HOME="/opt/android-sdk-linux"

export PATH="$ANDROID\_HOME/tools:$ANDROID\_HOME/platform-tools:$PATH"

// run this as Jenkins user

android update sdk --no-ui

Use *android list targets* to see what ABIs are installed. If no are listed manual install one.

### 8.4. Jenkins Plug-ins for installation

You need the follwing plug-ins to build Android application with Jenkins:

* Gradle Plugin
* Git Plugin - if Git is used for the project sources
* Android Emulator Plugin - supporting starting and unlocking an Android emulator and blocks the build until the emulator has started. Required for running tests. A detailed description of this plugin can be found under the following URL: [**Android Emulator Plugin Jenkins page**](https://wiki.jenkins-ci.org/display/JENKINS/Android+Emulator+Plugin).

### 8.5. Running an Android build with Jenkins

To create an Android build job on Jenkins, select New Job, enter a job name and select the Build a free-style software project option.

<< OLE Object: Picture (Device Independent Bitmap) >>

You configure from where the source should be cloned.

<< OLE Object: Picture (Device Independent Bitmap) >>

You configure the emulator which should be started. Ensure that you do not select the Show emulator window option, as your build server should not depend on the availability of a display server.

<< OLE Object: Picture (Device Independent Bitmap) >>

Add a Gradle build step to your Jenkins jobs.

<< OLE Object: Picture (Device Independent Bitmap) >>

### 8.6. Other useful plug-ins

The Android Emulator Plugin supports a new job with the Build multi-configuration project option. This option allows you to test multiple emulator configurations at the same time. You can, for example, test different languages, densities, screen resolutions, etc. Typically you have two Jobs, one for a simple build and test run and a multi-configuration project to test the build on different device configurations.

You can combine Android Emulator Plugin with the Amazon-EC2-Plug-in to run the build and theJenk tests on several machines simultaneously.

Another useful plug-in is the Lint plug-in which allows to run your Lint checks via Jenkins. See [**Lint Plug-in**](https://wiki.jenkins-ci.org/display/JENKINS/Android+Lint+Plugin) for details.

## 9. Jenkins backup and copying files

### 9.1. Jenkins backup and copying files

Jenkins stores all the settings, logs and build artifacts in its home directory, for example, in */var/lib/jenkins* under the default install location of Ubuntu.

To create a backup of your Jenkins setup, just copy this directory.

The *jobs* directory contains the individual jobs configured in the Jenkins install. You can move a job from one Jenkins installation to another by copying the corresponding job directory. You can also copy a job directory to clone a job or rename the directory.

Click reload config button in the Jenkins web user interface to force Jenkins to reload configuration from the disk.

See [**Adminstration of Jenkins**](https://wiki.jenkins-ci.org/display/JENKINS/Administering+Jenkins) for details.

### 9.2. Managing Jenkins with Git

Jenkins supports the <https://wiki.jenkins-ci.org/display/JENKINS/SCM+Sync+configuration+plugin> plug-in which allows you to store every change in a Git repo.

It is also possible to manually maintain the Jenkins configuration in a Git repo.

Please do n’ t circulate these case study anywhere. These are highly confidential for the GS account and GS academy . Please add email ids of GS pool resources, If I have missed any one names in the mailing list.

Regards,

Purushottam

<< OLE Object: Picture (Device Independent Bitmap) >>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, February 02, 2016 5:31 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry; Brij Bhushan Ahuja  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 4 (Devops and jenkins, ant , maven and Gradle)

Hi All,

Today we had detailed session via hands on session on Devops and Jenkins in 2 session (2hr in morning and 3 hr. in the evening ) .PPt for the sessions are attached as above.

<< File: Devops\_initial\_Understnading\_version\_Puru.pdf >> << File: jenkins\_Puru\_Final\_Version1.1.pptx.pdf >> .

Request you to go thu the ppt and revert me with queries that you have .

Agenda for tomorrow(10 to 12 pm):

We will discuss on :

* String
* Integer
* Sorting and
* Searching (If time permits).

Please do n’ t circulate these ppt’s. these are highly confidential for the GS account and GS academy . Please add email ids of GS pool resources. If I have missed any one names.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Thursday, January 28, 2016 10:11 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 2 (DS)  
**Importance:** High

Hi All , << File: DS4\_Queue\_GS\_Academy\_Puru.pdf >> << File: DS2\_LinkedList\_GS\_Academy\_Puru.pdf >>

Please find the day 2 ppt(Queue and linked List ) as an attachment . i am sharing the hands on assignments on queue and linked list as below :

1.Queue

<< OLE Object: Picture (Device Independent Bitmap) >>

2. Linked List

<< OLE Object: Picture (Device Independent Bitmap) >>

Please try to practice all these mentioned program on eclipse IDE .

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Thursday, January 28, 2016 4:59 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav; Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse; Anusha H Thimmappa; Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha; Ajay S Sastry  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 1 (DS)  
**Importance:** High

Hi All ,

Tomorrow we will be having DS session from 11 to 5 pm via LYNC. please use the same bridge that is mentioned in the mail chain.

Topic that we have covered till now are as below :

Day 1(27th Jan):DS

* 1.Datastruture and algorithms introduction (Finished on day 1 )
* 2.Stack ((Finished on day 1 )
* Ppt and assignments for hands on shared to the team on **Stack** (hands on assignment given to the team.)

Day2(28th Jan):DS

* Queue (Finished on day 2 )
* Linked List(Finished on day 2)
* I will be sharing the list of assignment and session ppt for the today’s session by tonight .

Day3:DS(Continued) :

Agenda for tomorrow (29th Jan )

* Time and space complexity
* Sorting.
* Searching(If time permits)

We will try to bind up the DS session tomorrow eod.

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, January 27, 2016 10:06 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav ([susheel.yadav@igate.com](mailto:susheel.yadav@igate.com)); Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse ([shahid.ghouse@igate.com](mailto:shahid.ghouse@igate.com)); Anusha H Thimmappa ([anusha.thimmappa@igate.com](mailto:anusha.thimmappa@igate.com)); Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha ([kundan.sinha@igate.com](mailto:kundan.sinha@igate.com)); Ajay S Sastry  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 1 (DS)  
**Importance:** High

Hi All ,

<< File: DS1\_DataStrctures\_Algorithams\_GS\_Academy\_Puru.pdf >> << File: DS3\_Stack\_GS\_Academy\_Puru.pdf >>

Please find the today’s session as an attachment.

Today’s assignment questions on stacks are as below :

<< OLE Object: Picture (Device Independent Bitmap) >>

It is suggested to write all the programs on the eclipse ide with proper logic .

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, January 27, 2016 5:31 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav ([susheel.yadav@igate.com](mailto:susheel.yadav@igate.com)); Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse ([shahid.ghouse@igate.com](mailto:shahid.ghouse@igate.com)); Anusha H Thimmappa ([anusha.thimmappa@igate.com](mailto:anusha.thimmappa@igate.com)); Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha ([kundan.sinha@igate.com](mailto:kundan.sinha@igate.com)); Ajay S Sastry  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** RE: GS Academy :Day 1 (DS)  
**Importance:** High

Dear All ,

It was nice interaction with you all after a long period of time.

# “The secret of getting ahead is getting started. The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and starting on the first one.”

― [**Mark Twain**](http://www.goodreads.com/author/show/1244.Mark_Twain)

Tomorrow we are planning the same from 11 to 5 pm on the LYNC please use the same bridge that is mentioned in the mail chain.

Topic we have covered today :

Day 1:DS

* **1.Datastruture and algoritham introduction**
* **2.Stack**

**I will be sharing the list of assignment and session ppt for the today’s session by tonight . It is highly suggested to complete all the assignment that we have discuss in the session and the assignment that we will be sharing you shortly.**

**Agenda for tomorrow (28th Jan )**

**Day2:DS(Continued)**

* **Time and space complecity**
* **Queue**
* **Linked List**
* **Sorting(If time permits)**

Regards,

Purushottam

-----Original Appointment-----  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, January 27, 2016 12:14 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav ([susheel.yadav@igate.com](mailto:susheel.yadav@igate.com)); Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse ([shahid.ghouse@igate.com](mailto:shahid.ghouse@igate.com)); Anusha H Thimmappa ([anusha.thimmappa@igate.com](mailto:anusha.thimmappa@igate.com)); Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha ([kundan.sinha@igate.com](mailto:kundan.sinha@igate.com)); Ajay S Sastry  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** GS Academy :Day 1 (DS)  
**When:** Wednesday, January 27, 2016 12:30 PM-1:00 PM (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi.  
**Where:** Lync Meeting  
**Importance:** High

HI Team ,

**Let’s meet today 3 to 5 pm for the 1st session on DS.**

.........................................................................................................................................

🡪 [Join Lync Meeting](https://meet.igate.com/purushottam.pukumar/MM39W9NG)

[Help](http://o15.officeredir.microsoft.com/r/rlidLync15?clid=1033&p1=5&p2=2009)

[!OC([1033])!]

.........................................................................................................................................

**Bridge details are as below :**

**Bridge details :**

|  |  |
| --- | --- |
| |  | | --- | | **Conference Code: 3057492   Leader Code: 1664281   (Do not Share with Participants)** | |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Instruction on how to use the bridge:** | | 1. Short dial code 2222 can be dialed from any IGATE office desk phone.For other countries refer table below  2. To start Conference as a Host, Please dial Leader code followed by # key on the welcome prompt.  3. To join conference as Participants, Please dial conference code followed by # key on the welcome prompt.  4. Hosts are not required to dial Conference code as it was the case in Cisco Meeting Place Audio Bridge (Erstwhile PMP or Patni Meeting Place). Hosts can directly enter Leader code after prompt and conference will start. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **IGATE Office phone Short Dial code** | **2222** | |  |  | | **Country** | **Toll Free Number(s)** | | USA | 18557257723 or 18554239839 or 18662570339 | | CANADA | 18552058340 or 18555616772 or 18662570339 | | UK | 8081683836 or 8004045899 | | INDIA | 18002001322 or 18001021400 | | SWITZERLAND | 0800836696 or 0800564188 | | NETHERLANDS | 8000234170 | | SINGAPORE | 8003211126 or 8001862804 | | GERMANY | 8001888047 | | FRANCE | 800913566 or 800913981 | |  |  | | **Country** | **Local Access number** | | USA | 19082198060 | | UK | 02082832301 | | INDIA,BANGALORE | 08030031300 or 08066600835 or 08041042222 | | INDIA,MUMBAI | 02225417522 | | INDIA,GANDHINAGAR | 07940016401 | | INDIA,NOIDA | 01207145205 | | INDIA,HYDERABAD | 04030874324 | | INDIA,PUNE | 02066853280 | | INDIA,CHENNAI | 04433032222 | | SWITZERLAND | +41435084553 | | NETHERLANDS | +31207956564 | | GERMANY | +4969257377301 | |  |  |   **Dial-in number web link:** | | <https://ispace.igate.com/Documents%20and%20Manuals/IT/IT%20Infrastructure/IGATE%20MX%20Audio%20Bridge%20Dial%20In%20Numbers.pdf>  **OR**  **Go to →** <https://ispace.igate.com> **→ Policies and Documents → IT → IT Infrastructure → IGATE MX Audio Bridge Dial in numbers** | |

**You can attend this from your desktop via Lync and login to the above metioned Bridge .**

Regards,

Purushottam

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, January 25, 2016 3:41 PM  
**To:** Kulachandramouli Vallapu; Gopi S Rao; Ashok Kumar(NABFS00); Vicky K Jaiswal; Neha Kumari; Srinivasa R Boddu; Yogendra G Ravva Venkata; Nayanjyoti Roy; Shabeena Khanum; Tejaswini Neeli; Susheel K Yadav ([susheel.yadav@igate.com](mailto:susheel.yadav@igate.com)); Mohamed Shiraz; SyedaTahaseen Naaz; Fazalulhaq Shaik; Shahid M Ghouse ([shahid.ghouse@igate.com](mailto:shahid.ghouse@igate.com)); Anusha H Thimmappa ([anusha.thimmappa@igate.com](mailto:anusha.thimmappa@igate.com)); Anil Kumar Sampson; Varada Prasanth; Pavithra V; Vasudev A Muddhan; Kundan K Sinha ([kundan.sinha@igate.com](mailto:kundan.sinha@igate.com)); Ajay S Sastry  
**Cc:** Satyaki Ranjan Ghosh; Arumugam Kanipakam; Sharad Khanna; Shrijith Shirnali  
**Subject:** GS Academy Study Materials Access@@@@  
**Importance:** High

Dear All ,

Please confirm , if you are able to access the GS Academy shared dive .

The path for shared drive is as below :

Path : [\\MUML11850\SharedGSAcademy](file:///\\MUML11850\SharedGSAcademy)

If you are not able to access the above mentioned path , please raise a Iconnect ticket (There might be Network  issue or DNS  issue ) immediate and get it resolved. We have given read access to all of you.

We are planning the training session from 27th Jan(Wednesday after 11 am).  we will be sharing the detailed content for day 1 (DS)  shortly .

Varada\Gopi : Please book a conf room till Friday(11 to 7 pm).

If you are able to access the shared drive . Please don’t circulate it . This is highly confidential for GS account and GS Academy . This is not for the distributions. We have seen couple of incident in the past. We have enabled the security for the same .

Regards,

Purushottam